ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

MARCH, 1983
Mike Dunn & Jim Bumpas, Editors

Ham/Hayes SmartModem Interface

by Burt Grebin (K2KLN) and Jerry Cudmore (WA2TLI)

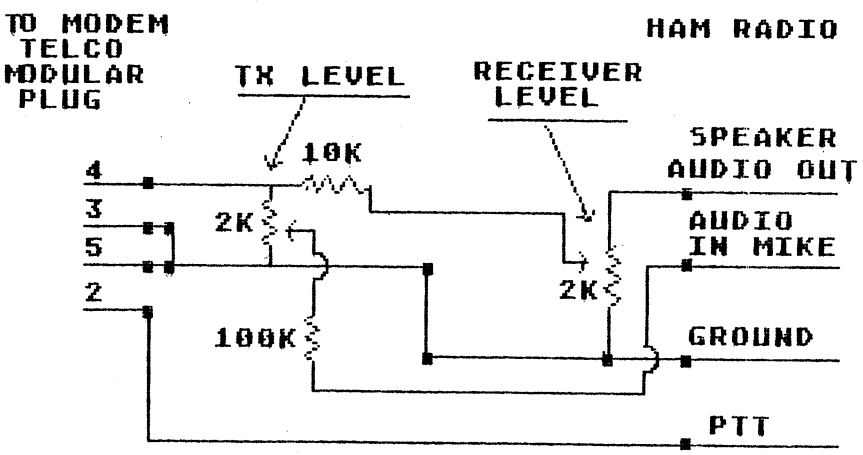

DIAGRAM BY GRAPHICS MASTER DUMPED TO A NEC PRINTER

News and Reviews

Mike Dunn, Co-editor

AtariWriter (Atari RX8036, under $100)

Several months ago, I was told by Gary Furr of Atari they had assembled a team to write a new word-processor combining the best features of all the current ones, be easy to use and well documented, and be on a ROM. The new AtariWriter, which I am using to write this article, is the result. It is easily the best piece of non-game software from Atari, and, with its low price, will be hard to beat.

Besides being on a ROM, it will support cassette as well as disk systems. It comes configured to use any of the Atari printers, and APX will sell printer drivers for popular printers when released in the spring.

You plug in the ROM, up comes the Atari logo, then the menu. You select from the following options: Create File, Delete File, Edit file, Format Disk, Index of disk files, Load File, Print File, or Save File. Starting a file, we choose Create, and the screen is shown. On top are the various format dispays for printing, and on the bottom are arrows showing the tab locations, as well as the line number and column number.

You can delete, move or repeat blocks of text, search and replace any string up to 25 charactors, both individually or globally. Centering, block text right, headers, footers, page numbering, and all the usual functions of a word processor are present, as well as some extra ones not always seen. You can go to the beginning or end of a line, use any printer control in the text by its CHR$ number, and do double column printing. You can even restore your last deleted text! Editing is done with an insert mode only, so you have to delete your old text- you can not write over it. Merging and chaining files can be done.

After you have finished, you can preview the text on the screen using your 40 col screen as a window to see a line up to 132 charactors wide. If you have specified it to be right justified, it will also be so on the screen.

Other features include the ability to edit any DOS file, and communications ability allowing owners of the AtariWriter to communicate with a Modem by using "R:" as the device name.

The program has excellent error trapping and prompting, making it almost impossible to lose files. The speed of accepting characters when editing, print formating, etc., is very impressive — it always keeps up with me, something previous word-processors have not always done.

The AtariWriter is an excellent product indeed, and very fairly priced. Since they were limited to a 16K ROM, they could not include everything. There are a few things I would like to have seen, but they are minor for most people. Having only an insert mode for editing and no type over mode means you have to erase everything to edit it. To be able to write your own printer drivers instead of having to get them from APX would be nice, but for most applications the ability to use printer control codes allows one to use most printers anyway.

A very fine product —look for it in April, with the printer drivers available at the same time.

—Mike Dunn

Inhome Keyboard

(Inhome software, 2485 Dunwin Dr., Mississauga, Ontario, Canada L5L 1T1, $120).

There are now a host of keyboards for the Atari 400. Because I wanted one which fits into the 400 rather than an outboard one, I bought the above. It is a full stroke keyboard which simply replaces the membrene keyboard on the 400. Very easy to install, and an improvement over the old one. Now the bad news — because it had to fit into the same space as the old one, there are some problems. The Return key is so far away I can't reach it. The shift key is frequently mixed up with the control key, and the backspace key is on the bottom of the keyboard — all of which makes it easy to mix up things. My main complaint, however, is the keyboard itself is very "tinny" and many of the keys stick. The picture in the advertisment shows the shift key to be different than the others, but mine isn't. It is a definite improvement over the membrane, but be sure to try it before you buy one.

ATR 8000 Disk Drive Controller

(Software Publishers, 2500 E. Randol Mill Rd., Arlington, TX 76011 (819)469-1181)

Always wanting to be one of the first to get a significant new piece of equipment for the Atari, I took the gamble and bought one of the first ones avaiable. This unit came with 64K, a parallel and serial port, a disk drive controller which will interface with single or double density drives, both 5 1/4"and 8", and runs CP/M. Since it is new, not all of the software is ready, and the hardware still has a few bugs in it. The company has been very helpful on the phone, and has given us much advice on getting it to run. The first problem was a tremendous amount of TVI that was helped by adding some capacitors. It has some trouble reading disks made on an Atari 810, but a new ROM is almost ready and is supposed to solve that problem. It is supposed to be able to read 5 1/4" Xerox 810 formated CP/M disks, as well as Kaypro with some modifications, and 8" single density CP/M disks. Software in the works now, to be sent free of charge to owners, include a high speed Atari disk copier, a CP/M modem program, a program to emulate various CP/M machines so their disks can be read, etc. The company will be coming out with a Newsletter to keep owners informed and, judging by the support they have given so far, expect it will be a good one.

When using it in the CP/M mode, the Atari becomes a dual 40 Col. terminal with 2 windows of 40 Col. each to match the 80 col needed for CP/M. You also have the choice of using a regular 80 col terminal, or, in the near future, a BIT-3 board. True 80 col. are needed for Word-Processors and other programs using cursor addressing.

Potentially a very nice unit, with a company trying very hard to make it all work ok. Right now, I recommend it only to those technically inclined who like to fiddle with and help debug new equipment. We will keep you posted, and, if there is enough interest, will have a reguler column for the ATR 8000.

DataSoft has just released their new **BASIC Compiler** for $70. **Empulse** (POB 593 Great Barrington, MA 01230) has sent a very nice cover for the Bulletin board computer. Microvector (132 s. Broad St. Canfield, OH 44406 suite B) sent a **Stick-Key** graphic key stickons for your keyboard for only $2.50.

COMPUTE! Books, Greensboro, NC has just released another fine book for the Atari, **Mapping the Atari** by Ian Chadwick. This is a very complete memory map and guide to every known memory location, with many examples and simple BASIC programs to illustrate their use, Over 194 pages, only $15.

We now have a local service station in Eugene, where we can get quality repairs quickly — many times in one day. A far cry from the 1 week and more when we sent it away! They do good work, and will repair out of area Atari equipment, warranty work, etc., for ACE members. **Mr. Television** 780 Blair Blvd, Eugene 503-485-4876.

Atari has announced their summer computer camps for all over the U.S. — New England, Poconos (PA), Chesapeake, Smokey Mountains, Midwest (Land of 10000 Lakes, Minnesota), Danville (CA), and San Diego. These have all the usual camp things plus computer training at all levels. These look like deluxe camps, so I expect they are not inexpensive. They also need Computer experts, counselors, teachers, directors, etc. For job information, send resumes to Atari Computer Camps, ATTN: Pat Tubbs, Dept F, 1196 Borregas, POB 427, Sunnyvale, CA 94086. For camp information, call 1-800-847-4180, or in NY or Canada, call collect 212-889-5200. I wish I could go!

—Mike Dunn

DATA PERFECT. . .AGAIN

For those of you who may wonder why it is you are now reading our third review of DATA PERFECT, I will explain. We of the ACE feel useable utility programs are extremely important. In my mind, the very existence of the personal (or home) computer is based on its ability to be used as a tool. At least that is why I purchased mine. A computer without worthwile utility programs available is no more than a very expensive toy.

In my last review, I said the programming behind DATA PERFECT is extremely well done, and the program presents more power and capability than most home users are apt to outgrow any time soon. My admiration for the skill and labor of the programmers was balanced by my dislike of the extremely poor documentation. As previously stated, the book stank. In that article I pleaded with LJK to hire a writer and redo the book. I don't know how much our review had to do with their actions, but they hired a writer and re-wrote the book from the ground up. Not only did they produce a complete re-write, but they did so in only 2 months.

In this age of unresponsive corporations and sellers of goods who care nothing for the needs or wants of the consumer, it is quite refreshing to find a company who listens. That they acted so quickly speaks of a real desire to provide the public with the products and support the users want and need. The NEW documentation which we received is as good as the original was bad.

Beginning with loading the disk, and explaining each step in a key-by-key manner, the new book takes the reader through building a sample data-base, which happens to be a check register. This data-base you create is useable and the report you create later is also the real thing. It can be used for your own records if you write enough checks to warrant computer time.

After following through this and the other examples in the book, you will find yourself familiar enough with the capabilities of DATA PERFECT to create your own custom data-base and begin to reap the rewards of computer record-keeping. Instead of the six weeks of study I recommended earlier, a normally intelligent person can become conversant with DATA PERFECT in a weekend.

When only the old documentation was available, my recommendation was for those of you whose needs were simple and/or who didn't want to spend six weeks studying, to buy FILE MANAGER 800+. Now that LJK has straightened out the documentation BUG, I can wholeheartedly recommend DATA PERFECT as the most powerful, flexible, and USEABLE data-base management program currently available for the ATARI personal computer.

To LJK, congratulations on a remarkably quick and competent recovery from a major blunder. Keep that new writer on your staff, he/she is GOOD. To SYNAPSE, how soon will we see a machine-language re-write of FILE MANAGER 800+ ?????

—Kirt E. Stockwell

HELP!

The documentation for Filemanager 800+ says up to 9 (!) pages of 20 fields each are available when creating forms. So far, I've been unable to use more than one page. Does anyone know how to create forms with more than one page of fields?

Over a month ago, I sent a SASE to SYNAPSE explaining my problem and asking for the solution. No answer. So I'm putting the question out to ACE readers.

If you can help, please respond by contacting: Jim Bumpas, Co-Editor of ACE, 4405 Dillard Road, Eugene, OR 97405 (503) 484-9925. Thanks.

LE Parallel Disk Drives

Here is the rest of the review I started in August. I'm late because the double density wasn't finished before now.

After ordering in May I got the single density drive two and a half months later. The last week before Christmas I got the double density version of the LE system (which runs the same as the Percom dd). The price of the complete one drive system is $1150.00. Each additional drive is $375.00.

You must have a monitor and an ATARI 800. The OS board in the ATARI is replaced by another board and this board extends four inches above the 800 and throws off enough RF to make it impossible to use any TV within 20 feet. I also highly recommend another brand of disk drive (i.e., Percom) as a back up. I have had mine go out many times. The weight of the LE system board causes the mother connections in the ATARI (which are not made to stand up to four times the weight and the added leverage) to be bent. This results in a poor connection, stopping operation. The extra weight causes the board to be unstable in the ATARI. I finaly had to put in my board guides to help hold it. The cure for this is to replace the 44 pin card connection in the bottom. It is very susceptible to dust and general corrosion of the contacts so you must clean them at least twice a month and sometimes daily if it is used in a dusty environment.

In the early systems, the board was a prototype. This means the board often has to be repaired or modified. Mine developed two broken traces which I spent some time on before tracking down and fixing it myself.

There seems to be confusion on how to offer the upgrades. Service takes a while. Usually two to three weeks and you have to send in the whole system. The manual was supposed to be modelled after the ATARI manuals, but it seems little more than a sales flyer. I wish they had gone into detail on how their system works, so as to use and address the six expansion slots. The information about memory locations is given a name but not equated! There are some other bugs which have been fixed. One bug destroyed some of the disk directory when the door was closed. I have lost a few non-replaceable disks this way.

So now the features it has permit complete control over the disk drive. The disk is controlled from the new board in the ATARI rather than from inside the drive like the 810 or Percom. It achieves some of its increased speed with staggered sectoring rather than using every ninth sector as all the others do. This allows you faster access time to the disk. There is also a speed increase from the parallel data transfer. All this means is that Programs like DOS boot in half the time and files like the ATARI Macro assembler load in 5 seconds vs. 15 seconds (Atari fast format). The disk drive mechanism is from Tandon and is rock steady with none of the timing problems.

Some of the things which can't be done on the 810 or a Percom (for more money of course; a copy type system runs $2800.00 for two drives): Bad sectoring; analog copying; deleting and reading of the sector marks which is vital for those producing software protection. These things can also be done (probably cheaper) by using a modified TRS-80 system (speed adjustment and custom software).

The 4k eprom (which occupies C000-D000 location in the ATARI) is what gives the LE system its parallel disk access. The LE system format of every other sector is great unless you try reading it on an ATARI 810 drive because it must wait for a whole track plus one sector, so it's slow.

They also have a 80 col text board (bit three board), an EPROM burner ($100), and a parallel printer interface. The promise of a CPM and winchester hard disk boards have been shelved. Until they redesign the LE system board to better accommodate the 800 it will still be only a prototype. Before you get into this, be sure you feel comfortable repairing and troubleshooting computers. The interface board has on it a Z80 processer running the disk interface with a 6520 and 1771.

Hayes Stack Smartmodem 1200

This is one of the new modems with a 1200 baud rate. The new breed of modems are only composed of a couple of chips, reducing the chance of something going out. The 1200 looks just like the 300 baud model and has features just like it. There is autodialing, redialing, originate, answer, and then all the settings for full/half duplex and parity. This is all done by sending a command over the modem when off-line. The modem intercepts all messages going out and looks for the special commands. You can do tone or pulse dialing. The 1200 stack has a built in speaker which monitors your dialing. A nice thing about it is the error codes it reports back to you about the state of the modem. The default state is 1200 baud.

Setting the stack up is easy: Plug it in and use a terminal program such as the APX Chameleon or the one from Swifty Software (Datalink). I use the APX version but it does have a bug which has been there for years (as long as it been out). This causes some confusion with the screen editor and the reading of the keyboard when doing a file transfer. But with the Chameleon source code all you do is change three lines in the assembly file of TSOP to read

```
1900 DOWN2 LDA CH
2000 CMP #255
2010 BEQ DOWN2
```

Then reassemble.

I have not used the Datalink but have heard good things about it (I don't know; it may not run at 1200 baud though). The list price of the 1200 Hayes Stack is $699.00 (often discounted to around $550). This is well worth it if you use your Atari as a terminal and if the other end has a 1200 hookup.

—George Shields, Spokane, WA

THE PRESIDENT'S STATE OF THE ONION ADDRESS

The goals of ACE are many-fold: First, to encourage the use pf computers in the home and the educational environment; Second, to develop the microcomputer (or its successor) as a viable and valuable tool in the home; Third, to make information and software available to all owners of Atari microcomputers as inexpensively as possible; Fourth, to encourage free enterprise by promoting the authoring of quality software for the Atari, and by strenuously discouraging software piracy and copyright infringement.

Subordinate to the goal of encouraging the use of microcomputers in the home and the educational environment, the Eugene, Oregon base group of the club has founded the ERACE (Educational Researchers of ACE).

This group will be instrumental in developing a comprehensive catalog of educational software, as well as writing specifications for needed programs to fill the gaps. The ERACE will also be developing what we hope will be the most complete and useful Program Review Form available.

There are other ACE groups supporting auxiliary programs in Eugene. FORTH: Exploring the language FORTH and developing a pool of competent FORTH programmers. This group is also working with the local APPLE people to determine the transportability of FORTH. BASIC: Teaching the BASIC language to members interested in learning. Also helping new members to become familiar with the equipment and overcome their fears. MACHINE LANGUAGE: Teaching the use of ASSEMBLER and what really goes on inside the computer. This group is currently led by two of our bright young mid-teenagers.

Within the next several months we expect to begin another group, this one devoted to the youth crowd. The age range is still up in the air, but this group will work with programming skills on a level most young people will be able to cope with. We look forward to getting this group fired up.

Another group, one which has just been formed, is the Exchange Library group. This group will help our overworked exchange librarian husband/wife team to reorganize and catalog the program library, as well as adding basic documentation to those programs needing it.

You will have noticed we carry no advertising in the newsletter. Our wish is to disperse as much information as possible within our budget. (The blurb about computerized typesetting nets us a hefty discount on our bill.)

When purchasing equipment for the club, we remember the majority of the funds available come from out of town, and we are as frugal as we can be. We make every attempt to see that the club membership in general derives full value from every dollar spent, rather than making things nice for us locals at the expense of all.

As the President, I want to say a few words about the people who actually run the club. (I just point the direction I think we should be going occasionally).

Many (most) of the organizations I have been involved with over the years have been very bottom-heavy. There have been far too many followers for the few chiefs to properly manage. Other groups seem to thrive on parliamentary procedure and useless formalities. I am proud to say this group has a larger number of truly capable and motivated leaders per capita than any other group I've seen. Their backgrounds run the gamut from doctors and lawyers to millworkers and students. None of them care what the occupation or social standing of the others, they are all united by a common purpose. The leaders of the ACE contribute not only many hours of actual "on-duty" time each month, but also field phone calls, work on their group projects, and work on their individual pet projects. I don't believe there is a finer group leading any club anywhere (or most corporations for that matter).

Now for the names of the COUNCIL:

Kirt E. Stockwell, PREZ
Larry Gold, VICE PREZ
Charlie Andrews, SEC/TREAS
Mike Dunn and Jim Bumpas, CO-EDITORS
Chuck and Jody Ross, EXCHANGE LIBRARIANS
Alice Miles Erickson, ERACE
Charlie Andrews, FORTH
Larry Gold, BASIC
John Atack and L.J. Knoll, MACHINE LANGUAGE

Other valuable persons are Ruth Elsworth, Ron Ness, DeLoy Graham, E.J. Knoll, and our PAST PRESIDENT and FOUNDER Stacy Goff. I could add another 20 or 30 names to this list, but space is getting tight.

Our goal here in Eugene is to serve the entire membership of the club. If for any reason you feel we are failing or fouling up, or if you want some help with a project, please contact us either through the mail, the phone, or the bulletin board. We want to do this right, and your feedback is valuable to us. One last thing. Any person or group wishing a copy of our CONSTITUTION can obtain one by sending a dollar to the exchange librarian to cover the cost of duplication and mailing.

Respectfully,

Kirt E. Stockwell
President

HAM ATARI

Having been an amateur radio operator since the age of eleven I have had the opportunity to experiment with many facets of this exciting communications hobby. For those of you who are not familiar with Ham radio, amateurs can communicate with one another using equipment capable of transmitting and receiving AM, FM, SSB (single sideband), NBFM (narrow band FM), RTTY (radio teletype), SSTV (slow scan TV), FSTV (fast scan), and CW (Morse code). In addition, amateur operators can utilize repeaters, transponders, and satellites to increase the range, effectiveness and reliability of their communications.

Since ham radio is my first love (after my wife and children — although they disagree at times) it didn't take long before I begin looking for a suitable match between my Atari 800 and my radio equipment. Unfortunately, the local hams I know either aren't interested in computers or don't know anything about the Atari. So, for a long time I had to be content with my lot and somewhat satisfied satisfied my desire to communicate via computer by frequenting a number of BBS across the country. I have met some very interesting people, but that certain "something"I enjoy with amateur radio is missing. To make matters worse, all the major ham publications print articles on computer networks, on-the-air nets and technical descriptions about interfacing radio equipment with computers. That was it! I called my good friend and fellow amateur, Jerry WA2TLI, and told him he had to buy an Atari 800 so we can experiment. After some heavy discussions with Jerry's wife she was convinced he definitely needs to be involved with computers. We each bought Hayes Smartmodems and then the fun began.

In the beginning Jerry and I uploaded and downloaded over the telephone using Datalink from Swifty Software. Each month our phone bills increased and we quickly decided there has to be a better way. Indeed there is. Right there in the Smartmodem owner's manual is the documentation we need to adapt the modem for use with radio equipment. Fortunately, the FCC recently approved amateur use of ASCII computer codes so all we had to do was build an interface (terminal unit). I'll get into the details in a moment, but first a brief discussion of the theory is in order.

When we transmit RTTY on VHF frequencies we are sending two different audio tones to represent the MARK and SPACE RTTY conditions. The MARK is a "rest"machine condition with the current ON. The SPACE condition has current OFF. We use these tones to directly modulate the FM transmitter to produce the F2 emission (audio tone modulation of a frequency modulated signal). Two different tone frequencies are used for the RTTY MARK and SPACE, respectively. The difference between them is called the RTTY Shift Frequency, or for our purposes in this discussion, AFSK (Audio Frequency Shift Keying). To summarize, two electronic interfaces are required for AFSK: A tone encoder to convert the mark and space pulses into audio tones during transmission; and a tone decoder to convert the pulses back during reception. The Modem or Terminal Unit (TU) is the device which combines the transmit tone encoder with the receive tone decoder.

The Smartmodem is ideally suited to handle telephone line and amateur radio data communications at rates up to 300 baud. It contains both tone encode and decode circuitry, permits automatic handshaking between data terminals, includes touch-tone dialing, and is designed to operate in either half or full duplex operation. Although full duplex operation is preferable (data or text can be received and printed while we are entering and transmitting data), most amateur communications take place in half duplex. In other words, we take turns talking or sending and receiving data.

With this general explanation of the theory behind the operation, let's move to the construction of an interface between the Smartmodem and the amateur transmit/receive equipment together with data codes to set the modem to half duplex.

Configure the dip switches behind the front panel of the Smartmodem as follows:

1,2,4,6—UP
3,5,7,8—DOWN

NOTE: Switch #7 must be down in order to key the transmitter.

Use the following keyboard codes to TRANSMIT data:

ATM0 - mutes the modem speaker (optional)
ATF0 - sets the modem to half duplex
ATS10 = 255A - keys the transmitter and sends the carrier tone

NOTE: Set the transmit deviation to +/-3KHz.

Use the following keyboard code to RECEIVE data:

ATC0H2D

NOTE: When setting up for receive, raise the volume on the receiver until a "Connect" is displayed on the screen. Then reduce the volume until a "No Carrier" appears. Proper setting for receive volume is just past this point.

This is all there is to it! Just arrange ahead of time who will originate (transmit) and who will answer (receive). Of course, if both of you can talk to each other on another frequency, then all non-data communication can take place there.

If this article stimulates interest let me know and I will encourage Jerry to write about full duplex operation, RTTY on VHF and UHF FM repeaters, RTTY on simplex channels using the Kantronics Interface and data transferring using SSB (single sideband) on the HF (high frequency) bands.

Happy ham-computing!!!

—Burt Grebin - K2KLN

"A SONG IS A WONDERFUL KIND OF THING"

Using Sound in ATARI PILOT To Encourage Children To Program
by Ruth Ellsworth

It is often difficult to get children to take the time to actually sit down and program. Sound in ATARI PILOT is easy enough that, with a little encouragement, even elementary age children can be helped to experiment with writing programs. This is possible because music in ATARI PILOT requires the mastery of only two commands: the SO: command, and the PA: command.

The SO: command plays the sound (note) desired and can hold up to four voices (variables). The variables for the notes Low C to High F# are the numbers 1 through 31. Thus a C major tonic chord is programmed SO:13,17,20 (13 is the variable for Middle C).

The PA: command determines the duration of the note desired (technically it pauses the designated length of time until the next command is processed). It follows the SO: command for the note desired and can have a value from 0 to 3600. The PA: command for the notes usually associated with music are: Whole Note = PA:128, Half Note = PA:64, Quarter Note = PA:32, Eighth Note = PA:16, Sixteenth Note = PA:8, and Thirty Second Note = PA:4. These values are relative as the tempo of music varies. Therefore, children can be encouraged to experiment with the sound of music and tempo as well as programming through the use of ATARI PILOT.

An understanding of the value relationships of the different musical notes can be helpful to a child when programming in ATARI PILOT, but is not essential. Experimenting with musical programming seems to help children acquire an understanding of those relationships as they develop skill and tone recognition.

The program which follows is written with two purposes in mind. The first is to allow our children to "play" the computer as they would the piano, and even our just turned three year old likes "to make the computer sing." The second purpose is to help them associate the SO: commands of PILOT with the keys of the piano.

In the text area of the screen the name of the piano keyboard key is given, the letter on the computer keyboard to be typed to "play" the note, and the SO: value of the note (space limitations do not allow those values to be displayed as attractively as desired, but work when explained to the child). Each note will play until the next note is typed or the space bar is pressed. The space bar has the SO: value of 0 so the music can be stopped at any time, and to allow a note to be played repeatedly, yet separately.

Because of the required listing space, I have not included sharps or flats in this routine. It is a "C Major Player Piano" as my children might say. That routine and a routine to save the variables to disk or cassette will be included in our #2 Pilot Disk which we will be giving to ACE this spring.

The listing itself is quite straight-forward. ATARI PILOT runs fastest when the modules used most often appear at the beginning of the listing because PILOT searches for modules each time from the top of the listing down. Therefore, the *SOUND module and note sounds appear first in an attempt to make the program run as rapidly as possible. The fact that ATARI PILOT is not a fast language is also the reason I did not change the key color as well as the note color.

The *SOUND module @B764 tells the computer to check the decimal value of the last key pressed and to jump to the module indicated if the listed key was pressed. No return key needs to be typed. Decimal values for the computer keyboard are found on page 50 of the techincal manual or can be found using the T: command. (Control characters are equal to the letter or number value of the key plus 128 to allow the use of control keys in programming.)

If you are typing this program in, I suggest you type the modules separately and save them. This is the way we build our programs which allows us to reuse the modules in many different ways by loading them then using the REN command. For example, this program which, draws a piano keyboard, staff, and notes, when loaded without the *SOUND module works nicely for games or review of the scale and its relationship to the keys of the piano, or the relationship of the notes on the staff. Typing programs by modules also makes debugging much easier because of the shorter listings. Typing can be kept to a minimum in this program because the lines tend to be similar. The lines in each module can be easily modified as needed by typing in the first line and then changing only the necessary characters. Line 100 becomes line 110 simply by moving the cursor over the first 0 and typing 1, moving the cursor to the value 22 and typing 18, then moving the cursor to X and typing C.

Programming music in ATARI PILOT can be "a song," and the means of encouraging children to become involved progarmming and experimenting with music.

RUTH'S PILOT

```
10 U:*BEGIN
20 U:*STAFF
30 U:*TREBLE
40 U:*BASS
50 U:*NOTES
60 U:*NAMES
70 U:*SOUND
80 E:
90 *SOUND
100 J(@B764=33):*NSD
110 J(@B764=22):*X
120 J(@B764=18):*C
130 J(@B764=16):*V
140 J(@B764=63):*A
150 J(@B764=62):*S
160 J(@B764=58):*D
170 J(@B764=56):*F
180 J(@B764=61):*G
190 J(@B764=57):*H
200 J(@B764=1):*J
210 J(@B764=5):*K
220 J(@B764=0):*L
230 J(@B764=45):*T
240 J(@B764=43):*Y
250 J(@B764=11):*U
260 J(@B764=13):*I
270 J(@B764=8):*O
280 J(@B764=10):*P
290 J:*SOUND
300 E:
310 *NSD
320 SO:0
330 *NNT
340 J(@B764=33):*NNT
350 J(@B764()33):*ND
360 E:
370 *ND
380 J:*SOUND
390 E:
400 *X
410 GR:PENRED;GOTO-50,2;4(DRAW1;TURN90
);GOTO-71,-27
420 SO:1
430 *XNT
440 J(@B764=22):*XNT
450 J(@B764()22):*XND
460 E:
470 *XND
480 GR:PENERASE;GOTO-50,2;4(DRAW1;TURN
90);GOTO-71,-27;PENBLUE;GOTO-50,2;4(DR
AW1;TURN90);GOTO-71,-27
490 J:*SOUND
500 E:
510 *C
520 GR:PENRED;GOTO-43,5;4(DRAW1;TURN90
);GOTO-63,-27
530 SO:3
540 *CNT
550 J(@B764=18):*CNT
560 J(@B764()18):*CND
570 E:
580 *CND
590 GR:PENERASE;GOTO-43,5;4(DRAW1;TURN
90);GOTO-63,-27;PENBLUE;GOTO-43,5;4(DR
AW1;TURN90);GOTO-63,-27
600 J:*SOUND
610 E:
620 *V
630 GR:PENRED;GOTO-36,7;4(DRAW1;TURN90
);GOTO-55,-27
640 SO:5
650 *VNT
660 J(@B764=16):*VNT
670 J(@B764()16):*VND
680 E:
690 *VND
700 GR:PENERASE;GOTO-36,7;4(DRAW1;TURN
90);GOTO-55,-27;PENBLUE;GOTO-36,7;4(DR
AW1;TURN90);GOTO-55,-27
710 SO:0
720 J:*SOUND
730 E:
740 *A
750 GR:PENRED;GOTO-29,10;4(DRAW1;TURN9
0);GOTO-47,-27
760 SO:6
770 *ANT
780 J(@B764=63):*ANT
790 J(@B764()63):*AND
800 E:
810 *AND
820 GR:PENERASE;GOTO-29,10;4(DRAW1;TUR
N90);GOTO-47,-27;PENBLUE;GOTO-29,10;4(
DRAW1;TURN90);GOTO-47,-27
830 J:*SOUND
840 E:
850 *S
860 GR:PENRED;GOTO-22,12;4(DRAW1;TURN9
0);GOTO-39,-27
870 SO:8
880 *SNT
890 J(@B764=62):*SNT
900 J(@B764()62):*SND
910 E:
920 *SND
930 GR:PENERASE;GOTO-22,12;4(DRAW1;TUR
N90);GOTO-39,-27;PENBLUE;GOTO-22,12;4(
DRAW1;TURN90);GOTO-39,-27
940 GR:PENBLUE;GOTO-22,12;4(DRAW1;TURN
90)
950 J:*SOUND
960 E:
970 *D
980 GR:PENRED;GOTO-15,15;4(DRAW1;TURN9
0);GOTO-31,-27
990 *DSO
1000 SO:10
1010 *DNT
1020 J(@B764=58):*DNT
1030 J(@B764()58):*DND
1040 E:
1050 *DND
1060 GR:PENERASE;GOTO-15,15;4(DRAW1;TU
RN90);GOTO-31,-27;PENBLUE;GOTO-15,15;4
(DRAW1;TURN90);GOTO-31,-27
1070 J:*SOUND
1080 E:
1090 *F
1100 GR:PENRED;GOTO-8,17;4(DRAW1;TURN9
0);GOTO-23,-27
1110 SO:12
1120 *FNT
1130 J(@B764=56):*FNT
1140 J(@B764()56):*FND
1150 E:
1160 *FND
1170 GR:PENERASE;GOTO-8,17;4(DRAW1;TUR
N90);GOTO-23,-27;PENBLUE;GOTO-8,17;4(D
RAW1;TURN90);GOTO-23,-27
1180 J:*SOUND
1190 E:
1200 *G
1210 GR:PENRED;GOTO-1,20;4(DRAW1;TURN9
0);GOTO-15,-27
1220 SO:13
1230 *GNT
1240 J(@B764=61):*GNT
1250 J(@B764()61):*GND
1260 E:
1270 *GND
1280 GR:PENERASE;GOTO-1,20;4(DRAW1;TUR
N90);GOTO-15,-27;PENBLUE;GOTO-1,20;4(D
RAW1;TURN90);GOTO-15,-27
1290 J:*SOUND
1300 E:
1310 *H
1320 GR:PENRED;GOTO6,22;4(DRAW1;TURN90
)
1330 GR:GOTO-7,-27
1340 SO:15
1350 *HNT
1360 J(@B764=57):*HNT
1370 J(@B764()57):*HND
1380 E:
1390 *HND
1400 GR:PENERASE;GOTO6,22;4(DRAW1;TURN
90);GOTO-7,-27;PENBLUE;GOTO6,22;4(DRAW
1;TURN90);GOTO-7,-27
1410 J:*SOUND
1420 E:
1430 *J
1440 GR:PENRED;GOTO13,25;4(DRAW1;TURN9
0);GOTO1,-27
1450 SO:17
1460 *JNT
1470 J(@B764=1):*JNT
1480 J(@B764()1):*JND
1490 E:
1500 *JND
1510 GR:PENERASE;GOTO13,25;4(DRAW1;TUR
N90);GOTO1,-27;PENBLUE;GOTO13,25;4(DRA
W1;TURN90);GOTO1,-27
1520 J:*SOUND
1530 E:
1540 *K
1550 GR:PENRED;GOTO20,27;4(DRAW1;TURN9
0);GOTO9,-27
1560 SO:18
1570 *KSO
1580 J(@B764=5):*KSO
1590 J(@B764()5):*KND
1600 E:
1610 *KND
1620 GR:PENERASE;GOTO20,27;4(DRAW1;TUR
N90);GOTO9,-27;PENBLUE;GOTO20,27;4(DRA
W1;TURN90);GOTO9,-27
1630 J:*SOUND
1640 E:
1650 *L
1660 GR:PENRED;GOTO27,30;4(DRAW1;TURN9
0);GOTO17,-27
1670 SO:20
1680 *LSO
1690 J(@B764=0):*LSO
1700 J(@B764()0):*LND
1710 E:
1720 *LND
1730 GR:PENERASE;GOTO27,30;4(DRAW1;TUR
N90);GOTO17,-27;PENBLUE;GOTO27,30;4(DR
AW1;TURN90);GOTO17,-27
1740 J:*SOUND
1750 E:
1760 *T
```

```
1770 GR:PENRED;GOTO34,32;4(DRAW1;TURN9
0);GOTO25,-27
1780 SO:22
1790 *TSO
1800 J(@B764=45):*TSO
1810 J(@B764()45):*TND
1820 E:
1830 *TND
1840 GR:PENERASE;GOTO34,32;4(DRAW1;TUR
N90);GOTO25,-27;PENBLUE;GOTO34,32;4(DR
AW1;TURN90);GOTO25,-27
1850 J:*SOUND
1860 E:
1870 *Y
1880 GR:PENRED;GOTO41,35;4(DRAW1;TURN9
0);GOTO33,-27
1890 SO:24
1900 *YSO
1910 J(@B764=43):*YSO
1920 J(@B764()43):*YND
1930 E:
1940 *YND
1950 GR:PENERASE;GOTO41,35;4(DRAW1;TUR
N90);GOTO33,-27;PENBLUE;GOTO41,35;4(DR
AW1;TURN90);GOTO33,-27
1960 J:*SOUND
1970 E:
1980 *U
1990 GR:PENRED;GOTO48,37;4(DRAW1;TURN9
0);GOTO41,-27
2000 SO:25
2010 *USO
2020 J(@B764=11):*USO
2030 J(@B764()11):*UND
2040 E:
2050 *UND
2060 GR:PENERASE;GOTO48,37;4(DRAW1;TUR
N90);GOTO41,-27;PENBLUE;GOTO48,37;4(DR
AW1;TURN90);GOTO41,-27
2070 J:*SOUND
2080 E:
2090 *I
2100 GR:PENRED;GOTO55,40;4(DRAW1;TURN9
0);GOTO49,-27
2110 SO:27
2120 *ISO
2130 J(@B764=13):*ISO
2140 J(@B764()13):*IND
2150 E:
2160 *IND
2170 GR:PENERASE;GOTO55,40;4(DRAW1;TUR
N90);GOTO49,-27;PENBLUE;GOTO55,40;4(DR
AW1;TURN90);GOTO49,-27
2180 J:*SOUND
2190 E:
2200 *O
2210 GR:PENRED;GOTO62,42;4(DRAW1;TURN9
0);GOTO57,-27
2220 SO:29
2230 *OSO
2240 J(@B764=8):*OSO
2250 J(@B764()8):*OND
2260 E:
2270 *OND
2280 GR:PENERASE;GOTO62,42;4(DRAW1;TUR
N90);GOTO57,-27;PENBLUE;GOTO62,42;4(DR
AW1;TURN90);GOTO57,-27
2290 J:*SOUND
2300 E:
2310 *P
2320 GR:PENRED;GOTO69,45;4(DRAW1;TURN9
0);GOTO65,-27
2330 SO:31
2340 *PSO
2350 J(@B764=10):*PSO
2360 J(@B764()10):*PND
2370 E:
2380 *PND
2390 GR:PENERASE;GOTO69,45;4(DRAW1;TUR
N90);GOTO65,-27;PENBLUE;GOTO69,45;4(DR
AW1;TURN90);GOTO65,-27
2400 J:*SOUND
2410 E:
2420 *NOTES
2430 GR:GOTO-50,2;4(DRAW1;TURN90)
2440 GR:GOTO-43,5;4(DRAW1;TURN90)
2450 GR:GOTO-36,7;4(DRAW1;TURN90)
2460 GR:GOTO-29,10;4(DRAW1;TURN90)
2470 GR:GOTO-22,12;4(DRAW1;TURN90)
2480 GR:GOTO-15,15;4(DRAW1;TURN90)
2490 GR:GOTO-8,17;4(DRAW1;TURN90)
2500 GR:GOTO-1,20;4(DRAW1;TURN90)
2510 GR:GOTO6,22;4(DRAW1;TURN90)
2520 GR:GOTO13,25;4(DRAW1;TURN90)
2530 GR:GOTO20,27;4(DRAW1;TURN90)
2540 GR:GOTO27,30;4(DRAW1;TURN90)
2550 GR:GOTO34,32;4(DRAW1;TURN90)
2560 GR:GOTO41,35;4(DRAW1;TURN90)
2570 GR:GOTO48,37;4(DRAW1;TURN90)
2580 GR:GOTO55,40;4(DRAW1;TURN90)
2590 GR:GOTO62,42;4(DRAW1;TURN90)
2600 GR:GOTO69,45;4(DRAW1;TURN90)
2610 E:
2620 *BASS
2630 GR:GOTO-65,10;4(DRAW1;TURN-90)
2640 GR:TURNTO180
2650 GR:5(DRAW1;TURN30)
2660 GR:9(DRAW1;TURN15)
2670 GR:18(DRAW1;TURN10)
2680 E:
2690 *TREBLE
2700 GR:PENBLUE
2710 GR:GOTO-65,19
2720 GR:TURNTO0;DRAW26;TURNTO90
2730 GR:9(DRAW1;TURN20)
2740 GR:21(DRAW1;TURN-8)
2750 GR:18(DRAW1;TURN-10)
2760 GR:5(DRAW1;TURN-20)
2770 GR:5(DRAW1;TURN-5)
2780 GR:18(DRAW1;TURN-30)
2790 GR:GOTO-65,19;4(DRAW1;TURN90)
2800 E:
2810 *STAFF
2820 GR:GOTO-75,45
2830 GR:DRAWTO75,45
2840 GR:GOTO-75,40;DRAWTO75,40
2850 GR:GOTO-75,35;DRAWTO75,35
2860 GR:GOTO-75,30;DRAWTO75,30
2870 GR:GOTO-75,25;DRAWTO75,25
2880 GR:GOTO-75,20;TURNTO90
2890 GR:16(DRAW2;GO7);DRAW2
2900 GR:GOTO-75,15;DRAWTO75,15
2910 GR:GOTO-75,10;DRAWTO75,10
2920 GR:GOTO-75,5;DRAWTO75,5
2930 GR:GOTO-75,0;DRAWTO75,0
2940 E:
2950 *BEGIN
2960 R:PIANO COLORS
2970 GR:CLEAR
2980 C:@B712=128+0[PEN ERASE - BACKGRO
UND - DARK BLUE
2990 C:@B709=0+0 [PEN YELLOW - BLACK
3000 C:@B710=144+14 [PEN BLUE - WHITE
3010 C:@B708=70+0 [PEN RED - LIGHT RED
3020 *KEYBOARD
3030 GR:PENBLUE;GOTO-75,-5 [KEY 1
3040 U:*FIRSTKEY
3050 GR:PENYELLOW;GOTO-69,-5 [KEY 2
3060 U:*BLACKKEY
3070 GR:PENBLUE;GOTO-64,-5 [KEY 3
3080 U:*KEYBETWEEN
3090 GR:PENYELLOW;GOTO-61,-5[KEY 4
3100 U:*BLACKKEY
3110 GR:PENBLUE;GOTO-56,-5 [KEY 5
3120 U:*RIGHTSK
3130 GR:PENBLUE;GOTO-50,-5 [KEY 6
3140 U:*LEFTSK
3150 GR:PEN YELLOW;GOTO-45,-5 [KEY 7
3160 U:*BLACKKEY
3170 GR:PENBLUE;GOTO-40,-5
3180 U:*KEYBETWEEN
3190 GR:PEN YELLOW;GOTO-37,-5 [KEY 9
3200 U:*BLACKKEY
3210 GR:PEN BLUE;GOTO-32,-5 [KEY 10
3220 U:*KEYBETWEEN
3230 GR:PEN YELLOW;GOTO-29,-5 [KEY11
3240 U:*BLACKKEY
3250 GR:PEN BLUE;GOTO-24,-5 [KEY12
3260 U:*RIGHTSK
3270 GR:PEN BLUE;GOTO-18,-5 [KEY13
3280 U:*LEFTSK
3290 GR:PEN YELLOW;GOTO-13,-5 [KEY14
3300 U:*BLACKKEY
3310 GR:PEN BLUE;GOTO-8,-5 [KEY15
3320 U:*KEYBETWEEN
3330 GR:PEN YELLOW;GOTO-5,-5 [KEY16
3340 U:*BLACKKEY
3350 GR:PEN BLUE;GOTO0,-5 [KEY17
3360 U:*RIGHTSK
3370 GR:PEN BLUE;GOTO6,-5 [KEY18
3380 U:*LEFTSK
3390 GR:PEN YELLOW;GOTO11,-5 [
3400 U:*BLACKKEY
3410 GR:PENBLUE;GOTO16,-5 [
3420 U:*KEYBETWEEN
3430 GR:PENYELLOW;GOTO19,-5 [KEY21
3440 U:*BLACKKEY
3450 GR:PENBLUE;GOTO24,-5 [KEY2
3460 U:*KEYBETWEEN
3470 GR:PEN YELLOW;GOTO27,-5 [KEY23
3480 U:*BLACKKEY
3490 GR:PENBLUE;GOTO32,-5 [KEY24
3500 U:*RIGHTSK
3510 GR:PEN BLUE;GOTO38,-5 [KEY25
3520 U:*LEFTSK
3530 GR:PEN YELLOW;GOTO43,-5 [KEY26
3540 U:*BLACKKEY
3550 GR:PEN BLUE;GOTO48,-5 [KEY 27
3560 U:*KEYBETWEEN
3570 GR:PEN YELLOW;GOTO51,-5
3580 U:*BLACKKEY
3590 GR:PEN BLUE;GOTO56,-5 [KEY29
3600 U:*RIGHTSK
3610 GR:PEN BLUE;GOTO62,-5 [KEY 30
3620 U:*LEFTSK
3630 GR:PEN YELLOW;GOTO67,-5 [KEY31
3640 U:*BLACKKEY
3650 E:
3660 *FIRSTKEY
3670 GR:TURNTO90;DRAW5;TURN90;DRAW15;T
URNTO90;DRAW2;TURN90;DRAW10;TURN90;DRA
W7;TURNTO0
3680 GR:FILL25 [WHITE KEY LEFT STRAIGH
T EDGE
3690 E:
3700 *LEFTSK
```

```
3710 GR:TURNTO90;DRAW4;TURN90;DRAW15;T
URNTO90;DRAW2;TURN90;DRAW10;TURN90;DRA
W6;TURNTO0
3720 GR:FILL25 [WHITE KEY LEFT STRAIGH
T EDGE
3730 E:
3740 *RIGHTSK
3750 GR:TURNTO90;DRAW4;TURNTO180;DRAW2
5;TURN90
3760 GR:DRAW6;TURNTO0;FILL10;TURNTO90;
DRAW2;TURNTO0;FILL15[WHITE KEY STRAIGH
T WHITE EDGE
3770 E:
3780 *BLACKKEY
3790 GR:TURNTO90;DRAW4;TURN90;DRAW14;T
URN90;DRAW4;TURN90;FILL14[BLACK KEY
3800 E:
3810 *KEYBETWEEN
3820 GR:TURNTO90;DRAW2;TURN90;DRAW15;T
URNTO90;DRAW2;TURNTO180;DRAW10;TURN90
3830 GR:DRAW6;TURNTO0;FILL10;TURNTO90;
DRAW2;TURNTO0;FILL15[WHITE KEY BETWEEN
 BLACKEYS
3840 E:
3850 *NAMES
3860 T:C D E F G A B C D E F G A B C D
 E F
3870 T:X C V A S D F G H J K L T Y U I
 O P
3880 T:1 3 5 6 8 10121315171820222425 2
72930
3890 E:
```

TINY TEXT 1.1

Tiny Text was originally written for members of **ACE** to produce "machine readable"documents for publication. I first saw it in the November, 1982 newsletter, where it received an upgrade from Jim Carr. After using my school's rather limited word processor for the Apple, I wanted at least that much power at home. Since I do not own a disk drive, most of the word processors cannot work for me. Therefore, I dug into Tiny Text to see if I could add a few functions. For instructions on how to use the program, see the November ACE Newsletter. I will only talk about the various text-imbedded commands Tiny Text now sports.

CTRL-E: Ends Current Line
CTRL-I: Indents Next Line
CTRL-S: Does a Linefeed
CTRL-T: Tabs x Number of Spaces
CTRL-C: Sets Center Justify Mode
CTRL-P: Does a forms feed.

The following commands are added:
CTRL-F: Sets Fill Justify Mode
CTRL-R: Sets Right Justify Mode
CTRL-L: Sets Left Justify Mode (Default)
CTRL-M: Sets Left Margin (Sequence terminated with "!")
CTRL-G: Sets Line Length (Sequence terminated with "!")
CTRL-H: Direct Printer Commands Follow (Again, terminated with "!")

These new commands allow one to reset the margins from within the text. This allows simple placement of the return address in a letter, the offsetting of a quote in an essay, and so on. Imbedding printer commands allows one to (depending on the printer) set the line spacing, form fee, and perhaps form length. Using the above method causes a new line to be started, making it useless for underlining a single word, superscripting, etc. However, Tiny Text now also allows the imbedding of direct printer commands right in the line through either of the following methods. "ESC"control codes may be sent, including null characters. To produce an "ESC"character, simply type the ESC key twice. This allows super/subscripting, underlining, changing character fonts on the same line, etc. A few words of caution: Each time ESC is encountered the program will automatically make room for two more codes. This may necessitate the use of null (CTRL-,) characters to pad those 2-stroke control codes.

As well, while fill justifying, should an ESC be encountered near the end of a line, and then be left to be printed in the next line, a rather messy situation occurs. The solution: Pad the offending line (the one just before the ESC code) with spaces). Some experience will be the best explanation to this.

The other method is to imbed inverse video control characters which the printer recognizes. Unfortunately, this will not work in the fill justify mode.

To use the new commands which are delimited with the "!"character, first type the CTRL character which designates the function you wish to use, then type the string or values you wish to pass, and end the sequence with "!". for example, suppose you want to set the left margin to "20"halfway through your text. You type "CTRL-M20!".

Another added feature is the allowing of a zero form feed for those of us who have printers which can be programmed to skip at the end of each form.

Those of you who already have the Tiny Text listing from the November newsletter can save a lot of typing. The only changed lines are the following: 115, 716-719, 721-723, 730, 750, 752, 782, 815, 820, 850, 2200-3135. In addition, lines 610 and 670 are deleted. In order to save memory, you may wish to delete the REMs in these lines.

While Tiny Text is still no replacement for a full blown word processor, it can still satisfy the needs of many people, and, best of all, it is absolutely free.

If you continue to have troubles, don't hesitate to write me: Dale Lutz, Box 373, Holden, Alberta, Canada, T0B 2C0.

—Dale Lutz

BUMPAS REVIEWS

On January 18, 1983 the IRS finally recognized ACE is exempt from Federal income tax pursuant to I.R.C. Section 501(c)(7). We now must file tax returns on IRS form 990 by the 15th day of the 5th month after the end of our annual accounting period for any year in which our gross receipts normally exceed $10,000.

This exemption was granted upon application filed by filling out IRS form 1024, which you may obtain from any local IRS office. The form is simple to fill out. You don't need to worry too much about getting everything in the application they want to see. Most likely they will send you a letter asking for additional information anyway. I've never seen one go through without the IRS asking for additional information.

To file for exemption, you need to exist as a formed organization with organic documents in written form. You must provide the IRS with copies of these documents (Constitution, Articles of Incorporation, By-Laws, etc.). You are not required to have **all** these documents (we have only a Constitution). But they do want to see at least one founding document. If you have more than one, they want copies of them all.

I believe it may be possible for a non-profit computer organization to gain recognition as a charitable educational organization. This will qualify donations made by persons to the organizations as tax-deductible. ACE does not have this charitable status, and so donations made are not tax-deductible. The donations are income to ACE, but ACE will not have to pay income tax on this income. To gain charitable status I think the organization will have to refrain from officially being limited to a particular brand name of computer. The organization will have to be officially oriented more broadly to computers and computer education in general.

If you have any specific questions about what I've said here, or get hung up anywhere in the application process, we might be able to answer your questions. Don't hesitate to ask.

—Jim Bumpas
Co-editor and
Attorney for ACE

GORF REVIEW

Gorf, by Roklan ($34.95), is a very fast action game consisting of four screens: AstroBattles, Laser Attack, Space Warp and Flag Ship. A fifth called Galaxians in the Arcade version was left out. It went between Laser Attack and SpaceWarp.

AstroBattles is a typical Space Invaders type set up. You have a shield which disappears when you press the button, and reappears when you let go. You try to blast three rows of invaders.

The second, Laser Attack, has you in space, and you have to shoot two sets of five invaders. There is one red "GORF", as well as three yellow Kamikaze ships and one blue anti-matter laser-bearer per set.

In the next screen the ships stay in the middle until, one by one, they come circling out. There is a web-type series of lines which confuse you. You can only shoot them as they circle, and they try to bomb you until they escape.

Lastly there is the dreaded FLAGSHIP. You must blast holes in its shield and blow pieces off the ship until you can get the reactors.

It is very difficult, and true to the Arcade Game. I recommend it highly.

—Eric Wright

```
1 REM ********************************
      **       TINY TEXT 1.1        **
      **                            **
2 REM **  by Stan Ockers Sept. 81   **
      **  ACE Newsletter Nov.  81   **
      **                            **
3 REM **  Originally modified by    **
      **   Jim Carr 01 Oct. 82      **
      **  ACE Newsletter Nov.  82   **
4 REM **                            **
      **    Second upgrade by       **
      **   Dale Lutz 04 Jan. 83     **
5 REM ********************************
      ADD THE FOLLOWING LINES AND
      DELETE LINES 610 AND 670.
115 MODE=1
716 IF B=6 THEN MODE=3
717 IF B=18 THEN MODE=2
718 IF B=12 THEN MODE=1
719 IF B=3 THEN MODE=4
721 IF B=7 THEN 2200
722 IF B=13 THEN 2300
723 IF B=8 THEN 3100
730 IF T$(TP+1,TP+1)=" " THEN TP=TP+1:
P=P+1:REM CHECKS FOR SPACES IN THE BEG
INNING OF A LINE, NOT WHAT WE WANT
750 A=ASC(T$(TP,TP)):IF A(32 AND A()27
 AND A()0 THEN C=0:GOTO 780
752 IF A=27 THEN RL=RL+3:REM ADDS THRE
E TO LINE IN ORDER TO COMPENSATE FOR
THE UNPRINTED ESCAPE CODES
782 IF MODE=1 OR MODE=2 OR MODE=4 THEN
 A$=T$(P+1,TP-1):GOTO 810
815 IF OP=3 AND FF)0 THEN LINE=LINE+1:
IF LINE)(PS-FF) THEN LINE=1:FOR I=1 TO
 FF:LPRINT " ":NEXT I
820 SP=LM+(B=9)*IND+(B=20)*TAB+(MODE=4
)*(LL-LEN(A$))/2+(MODE=2)*(LL-LEN(A$))
:IF SP)100 THEN SP=100
850 IF FL THEN FOR A=1 TO 1000:NEXT A:
GOTO 500
2200 TP=TP+1:GOSUB 3000:LL=TEMP:GOTO 7
15
2300 TP=TP+1:GOSUB 3000:LM=TEMP:GOTO 7
15
3000 TEMP=0
3010 TEMP=TEMP*10+VAL(T$(TP,TP)):TP=TP
+1:IF T$(TP,TP)="!" THEN P=TP:RETURN
3020 GOTO 3010
3100 TP=TP+1:A$="":I=1
3110 IF T$(TP,TP)="!" THEN P=TP:GOTO 3
130
3120 A$(I)=T$(TP,TP):I=I+1:TP=TP+1:GOT
O 3110
3130 IF OP=3 THEN LPRINT A$
3135 GOTO 715
```

```
                    10 ; Fill the screen with A's
                    20 ;
                    30 ; Note: Locations used are defined
                    40 ; at the beginning of the program.
                    50 ;
    =0058           60 SAVMSC =  $58
    =00CB           70 INDLO =   $CB
                    80 ;
0000                90       *=  $0600
                    0100 ;
0600 68             0110     PLA         ; For USR function
0601 A204           0120     LDX #$04    ; 4 pages
0603 A558           0130     LDA SAVMSC  ; Lo byte screen pointer
0605 85CB           0140     STA INDLO   ; into indirect pointer
0607 A559           0150     LDA SAVMSC+1 ; Hi byte also
0609 85CC           0160     STA INDLO+1
060B A921           0170     LDA #$21    ; An 'A' in screen code
060D A000           0180     LDY #$00    ; Offset is initially zero
060F 91CB           0190 LOOP1 STA (INDLO),Y ; Put byte on screen
0611 C8             0200     INY         ; Next screen position
0612 D0FB           0210     BNE LOOP1   ; Until Y is zero again
0614 E6CC           0220     INC INDLO+1 ; Up one page
0616 CA             0230     DEX         ; Last page?
0617 D0F6           0240     BNE LOOP1   ; Not yet
0619 18             0250 LOOP2 CLC       ; Loop continually
061A 90FD           0260     BCC LOOP2   ; (to prevent screen clearing)
                    0270 ;            LISTING #1
                    0280 ;
                    0290 ;
                    0300 ;
                    0310 ; Move character set from ROM into RAM
                    0320 ;
    =00CB           0330 FROMPT = $CB
    =00CD           0340 TOPNT = $CD
    =006A           0350 RAMTOP = $6A
                    0360 ;
061C                0370     *=  600
                    0380 ;
0258 68             0390     PLA         ; USR again
0259 A900           0400     LDA #$00    ; Initialize Lo bytes of pointers
025B 85CB           0410     STA FROMPT
025D 85CD           0420     STA TOPNT
025F A9E0           0430     LDA #$E0    ; Char set is at $E000
0261 85CC           0440     STA FROMPT+1 ; Hi byte char. set
0263 A56A           0450     LDA RAMTOP  ; Lower RAMTOP 5 pages
0265 38             0460     SEC         ; (necessary for subtraction)
0266 E905           0470     SBC #$05
0268 856A           0480     STA RAMTOP  ; Area now reserved
026A 18             0490     CLC         ; Back up one page
026B 6901           0500     ADC #$01
026D 85CE           0510     STA TOPNT+1 ; Hi byte new char. set
026F A204           0520     LDX #$04    ; 4 pages
0271 A000           0530     LDY #$00    ; index initially zero
0273 B1CB           0540 FILL LDA (FROMPT),Y ; Byte from ROM
0275 91CD           0550     STA (TOPNT),Y ; into RAM
```

```
0277 C8       0560     INY          ; Next byte
0278 D0F9     0570     BNE FILL     ; Rest of page
027A E6CC     0580     INC FROMPT+1 ; Next page from
027C E6CE     0590     INC TOPNT+1  ; Next page to
027E CA       0600     DEX          ; Finished?
027F D0F2     0610     BNE FILL     ; Not yet
0281 60       0620     RTS
              0630 ;          LISTING #2
              0640 ;
              0650 ;
              0660 ;
              0670 ; ANSWER TO PROBLEM #5
              0680 ;
   =0278      0690 STICK0 = $0278
   =0054      0700 ROWCRS = $54
   =0055      0710 COLCRS = $55
   =0002      0720 LTLIM ="$02
   =0025      0730 RTLIM = $25
   =0002      0740 TOPLIM = $02
   =0015      0750 BOTLIM = $15
              0760 ;
0282          0770     *=  $0600
              0780 ;
0600 68       0790     PLA
0601 AD7802   0800     LDA STICK0
0604 C90E     0810     CMP #$0E     ; stick up?
0606 D00A     0820     BNE DN       ; no
0608 A654     0830     LDX ROWCRS   ; reached top?
060A E002     0840     CPX #TOPLIM
060C F004     0850     BEQ DN       ; ROWCRS = TOPLIM
060E 9002     0860     BCC DN       ; ROWCRS ( TOPLIM
0610 C654     0870     DEC ROWCRS   ; no, up one
0612 C90D     0880 DN  CMP #$0D     ; stick down?
0614 D008     0890     BNE LEFT     ; no
0616 A654     0900     LDX ROWCRS   ; reached bottom?
0618 E015     0910     CPX #BOTLIM
061A B002     0920     BCS LEFT     ; ROWCRS )= BOTLIM
061C E654     0930     INC ROWCRS   ; down one
061E C90B     0940 LEFT CMP #$0B    ; stick left?
0620 D00A     0950     BNE RIGHT    ; no
0622 A655     0960     LDX COLCRS   ; reached left side?
0624 E002     0970     CPX #LTLIM
0626 F004     0980     BEQ RIGHT    ; yes, skip
0628 9002     0990     BCC RIGHT    ; COLCRS ( LTLIM
062A C655     1000     DEC COLCRS   ; one right move
062C C907     1010 RIGHT CMP #$07   ; stick right?
062E D008     1020     BNE OUT      ; no
0630 A655     1030     LDX COLCRS   ; reached right side?
0632 E025     1040     CPX #RTLIM
0634 B002     1050     BCS OUT      ; COLCRS )= RTLIM
0636 E655     1060     INC COLCRS   ; one move right
0638 60       1070 OUT RTS          ; Back to Basic
```

```
SYMBOLS

=0015 BOTLIM     =0055 COLCRS     0612 DN
0273 FILL        =00CB FROMPT
=00CB INDLO       061E LEFT       060F LOOP1
0619 LOOP2       =0002 LTLIM
 0638 OUT         =006A RAMTOP    062C RIGHT
=0054 ROWCRS      =0025 RTLIM      .
=0058 SAVMSC      =0278 STICK0    =0002 TOPLIM
=00CD TOPNT                        .
                                   .
```

Machine Language Programming

by Stan Ockers

#4 Indirect Indexed Addressing

This session introduces a powerful addressing mode especially useful in moving lots of memory around. In the process we will learn how to write the program of problem #3 without having it modify itself.

Rather than refer to an address directly we are going to refer to it INDIRECTLY by refering to another address (two bytes actually). The two bytes making up the indirect address must be located in zero page. We will use $00CB and $00CC. To further complicate matters we will tack on an index which must be Y. The instruction written LDA ($00CB),Y reads 'load the accumulator indirect modified by Y' and has the following meaning: Get an address by using the number in $00CB as the low byte and $00CC in the high byte. Add to it the value of Y and use that address as the one from which to fetch a number and put it in the accumulator.

Why all this round-about-ness? The power comes because it's easy to modify the zero page location and not change our program while doing so. Consider listing #1 which is a reworking of our 'fill screen with A's' program. Use 'Hexpoke' (Dec.-Jan. issue) to try it. Notice that after setting up our indirect address with the location of the screen (upper left corner). We use Y to successively fill 256 bytes. The high order byte of our pointer is incremented and Y goes through another 256 iterations until 4 pages are filled.

There are eight instructions which use the indirect indexed addressing mode but the ones we will use most often are:

LDA (IND),Y with op-code B1

STA (IND),Y with op-code 91

Another good use for the indirect indexed instructions is in moving the character set from ROM to RAM. You must do this if you want to change some characters. The character set starts at $E000 and is 1K long (4 pages). One way to create room for the character set is to put it at the very top of memory and fool the Atari into thinking there is less memory than normal. This is necessary because the display data is normally placed highest in memory. In fact it's possible for display data to overide the top of memory in some cases.

Location $006A (106 decimal) tells the Atari how many pages of memory are available. If we drop this by 5 pages and then go back up one page to insert our character set, the set will start on a 1K boundary (a requirement), and space is available for any display overflow. You must remember to give a graphics mode command (e.g., GR. 0) after this process to re-establish the display list and data. Location 756 (decimal) must also be POKEd with the page location of the new character set after any graphics mode command, (POKE 756,PEEK(106) + 1). If any PM graphics work is done, remember that RAMTOP is no longer on a 1K boundary.

OCKERS' MOTIE

```
110 REM **     MOTIE    **
120 REM **  S.O. 2/83   **
130 REM ** FROM GAME BY **
140 REM ** MAC OGLESBY  **
150 REM ********************
155 REM ** THIS GAME WILL NOT FIT IN 1
6K **
160 DIM GM$(11),MM$(11),GN$(38),MN$(55
),XP$(11),YP$(11),I$(11),L$(49),W1$(21
),W2$(19),I1$(11),I2$(11)
170 REM Gardian offset into moves list
180 RESTORE 190:FOR J=1 TO 11:READ A:G
M$(J,J)=CHR$(A):NEXT J
190 DATA 1,5,9,13,17,20,26,29,32,36,38
200 REM Motie offset into moves list
210 RESTORE 220:FOR J=1 TO 11:READ A:M
M$(J,J)=CHR$(A):NEXT J
220 DATA 1,5,10,15,20,24,33,37,42,47,5
2
230 REM Allowable Gardian next moves
240 RESTORE 250:FOR J=1 TO 38:READ A:G
N$(J,J)=CHR$(A):NEXT J
250 DATA 1,2,3,11,2,4,5,11,1,3,5,11,2,
5,6,11,5,7,11,4,6,7,8,9,11,5,9,11,8,10
,11,7,9,10,11,8,10,11
260 REM X positions of moves
270 RESTORE 280:FOR J=1 TO 11:READ A:X
P$(J,J)=CHR$(A):NEXT J
280 DATA 20,15,20,25,15,20,25,15,20,25
,20
290 REM Y positions of moves
300 RESTORE 310:FOR J=1 TO 11:READ A:Y
P$(J,J)=CHR$(A):NEXT J
310 DATA 9,7,7,7,5,5,5,3,3,3,1
320 REM Allowable Motie next moves
330 RESTORE 340:FOR J=1 TO 55:READ A:M
N$(J,J)=CHR$(A):NEXT J
340 DATA 1,2,3,11,0,2,4,5,11,0,1,3,5,1
1,0,2,5,6,11,1,5,7,11,1,2,3,4,6,7,8,9,
11
350 DATA 3,5,9,11,4,5,8,10,11,5,7,9,10
,11,5,6,8,10,11,7,8,9,11
360 REM Warp subroutine #1
390 RESTORE 400:FOR J=1 TO 21:READ A:W
1$(J,J)=CHR$(A):NEXT J
400 DATA 104,162,0,172,193,2,189,194,2
,157,193,2,232,224,8,144,245,140,200,2
,96
410 REM Warp subroutine #2
420 RESTORE 430:FOR J=1 TO 19:READ A:W
2$(J,J)=CHR$(A):NEXT J
430 DATA 104,162,7,172,200,2,189,193,2
,157,194,2,202,16,247,140,193,2,96
440 REM Lanes
450 L$=" //:\\  .. .. :\\://: .. .. :/
/:\\: .. ..  \\://  "
460 REM Draw warp screen
470 GRAPHICS 10:R=INT(RND(0)*16)*16:RE
STORE 480:FOR I=1 TO 8:READ A:POKE 704
+I,A+R:NEXT I
480 DATA 2,4,6,8,6,4,2,2
490 Q=1:FOR I=0 TO 38:COLOR Q:X=I:Y=I*
2:PLOT X,Y:DRAWTO 79-X,Y:PLOT X,Y+1:DR
AWTO 79-X,Y+1:DRAWTO 79-X,190-Y
500 DRAWTO X,190-Y:PLOT 79-X,190-Y+1:D
RAWTO X,190-Y+1:DRAWTO X,Y:Q=Q+1:IF Q>
6 THEN Q=1
510 NEXT I
520 REM save pointers, lower memory
530 LOSAV=PEEK(560):HISAV=PEEK(561):PO
KE 106,PEEK(106)-32
540 GOSUB 1960:REM Change character se
t
550 GOSUB 1190:REM Directions-selectio
ns
555 REM I$ holds screen configuration
560 I$="gg#g#######"
570 REM random free position for Motie
580 R=INT(RND(0)*9)+2:IF R=3 THEN 580
590 I$(R+1,R+1)=CHR$(109):MOTN=R
600 CURN=5:LCOL=32:MCOL=182:GCOL=246:B
KCOL=9:TURN=0
610 GOSUB 1560:REM draw galactic map
620 REM Choose Guardian move
630 IF STRIG(0)=0 THEN 630
640 GOSUB 1780
650 IF Z()37 THEN 630
660 GNUM=CURN:POSITION X,Y:? #6;"%&":R
EM %& are in inverse char. line 660
670 IF STRIG(0)=0 THEN 670
680 MFLAG=1
690 GOSUB 1780:J=0
700 IF Z=165 THEN POSITION X,Y:? #6;"%
&":MFLAG=0:GOTO 630
710 IF Z()35 THEN 690
720 MFLAG=0
730 R=ASC(GM$(GNUM+1))
740 K=ASC(GN$(R+J)):IF K=11 THEN 690
750 IF CURN()K THEN J=J+1:GOTO 730
760 X=ASC(XP$(GNUM+1)):Y=ASC(YP$(GNUM+
1)):POSITION X,Y:? #6;"#$":I$(GNUM+1,G
NUM+1)="#"
770 X=ASC(XP$(CURN+1)):Y=ASC(YP$(CURN+
1)):POSITION X,Y:? #6;"%&":I$(CURN+1,C
URN+1)="g"
780 REM Check for Guardian win
790 POKE 77,0:IF I$="#g##mg#g###" OR I
$="###g#gm##g#" OR I$="#######gggm" TH
EN 1450
800 REM Check for 3rd repetition
810 IF I$()I2$ THEN REPT=0:GOTO 830
820 REPT=REPT+1:IF REPT=3 THEN 1490
830 I2$=I1$:I1$=I$
840 REM Warp and redraw map
850 GOSUB 1700:GOSUB 1560
860 IF PFLAG=1 THEN GOSUB 1100:GOTO 10
30
870 FOR J=1 TO 6:FOR K=15 TO 0 STEP -1
:POKE 711,MCOL+K:SOUND 0,230+K,10,10:S
OUND 1,240-K,10,10:FOR L=1 TO 3
880 NEXT L:NEXT K:NEXT J:SOUND 0,0,0,0
:SOUND 1,0,0,0
890 REM Computer chooses Motie move
900 J=INT(RND(0)*3):R=ASC(MM$(MOTN+1))
910 K=ASC(MN$(R+J)):IF K=11 THEN J=0:G
OTO 910
920 IF I$="####gg#m#g#" THEN K=10
930 IF I$="#####ggg#m#" THEN K=10
940 IF I$="#####ggg##m" THEN K=9
950 IF I$="###g#g##gm#" THEN K=10
960 IF I$="#g###g#mg##" THEN K=10
970 IF MOTN=1 AND I$(1,1)="#" THEN K=0
980 IF MOTN=2 AND I$(1,1)="#" THEN K=0
990 IF MOTN=3 AND I$(1,1)="#" THEN K=0
1000 IF I$="######gggm#" THEN K=5
1010 IF I$="####g##mgg#" THEN K=5
1020 X=ASC(XP$(K+1)):Y=ASC(YP$(K+1)):L
OCATE X,Y,Z:IF Z()35 THEN J=J+1:GOTO 9
10
1030 POSITION X,Y:? #6;"'(":X=ASC(XP$(
MOTN+1)):Y=ASC(YP$(MOTN+1)):POSITION X
,Y:? #6;"#$":REM '( ARE IN INVERSE
1040 I$(MOTN+1,MOTN+1)="#":I$(K+1,K+1)
="m":MOTN=K
1050 REM Check for Motie win
1060 IF MOTN=0 THEN GOTO 1490
1070 TURN=TURN+1:IF TURN)14 THEN 1490
1080 POSITION 25,11:? #6;15-TURN;" TUR
NS LEFT "
1090 GOTO 630
1100 IF STRIG(0)=0 THEN 1100
1110 POSITION 25,11:? #6;" Motie turn"
:MFLAG=2:CURN=MOTN
1120 GOSUB 1780:J=0
1130 IF Z()35 THEN 1120
1140 R=ASC(MM$(MOTN+1))
1150 K=ASC(MN$(R+J)):IF K=11 THEN 1120
1160 IF CURN()K THEN J=J+1:GOTO 1150
1170 RETURN
1180 REM The 'ot' in line 1190 are in
inverse
1190 PFLAG=0:GRAPHICS 18:POSITION 7,4:
? #6;"Motie":POSITION 2,6:? #6;"SELECT
-1 Player ":POSITION 0,8
1200 POSITION 1,8:? #6;"OPTION-instruc
tions":POSITION 7,10:? #6;"START"
1210 FOR J=1 TO 30:NEXT J:POKE 53279,8
:IF PEEK(53279)=3 THEN GOTO 1260
1220 IF PEEK(53279)=5 AND PFLAG=0 THEN
 PFLAG=1:POSITION 9,6:? #6;"2 Players"
:GOTO 1210
1230 IF PEEK(53279)=5 AND PFLAG=1 THEN
 PFLAG=0:POSITION 9,6:? #6;"1 Player "
1240 IF PEEK(53279)=6 THEN RETURN
1250 GOTO 1210
1260 GRAPHICS 0:POKE 752,1:? :? :? "
  3 Guardian starships protect the"
1270 ? "Human Empire from attack by a
Motie"
1280 ? "vessel.  If there is only one
player"
1290 ? "the computer moves the Motie s
hip."
1300 ? "Two players alternate using st
ick #1."
1310 ? :? "    The Guardians can't mov
e"
1320 ? "downscreen while the Motie ves
sel"
1330 ? "can move any direction along t
he"
1340 ? "indicated starship lanes."
1350 ? "Moves are to adjacent empty po
sitions"
1360 ? :? "   Guardians win if the Mot
ie is"
1370 ? "trapped (has no legal moves)."
1380 ? "The Moties win if 15 turns hav
e"
1390 ? "elapsed, if there is a 3-time"
1400 ? "repetition of position or if t
heir"
1410 ? "vessel reaches the lowest posi
tion. "
1420 ? :? "PRESS KEY TO START":POKE 76
4,255
1430 IF PEEK(764)=255 THEN 1430
1440 GOTO 1190
1450 GRAPHICS 18:POSITION 2,2:? #6;"Th
e Motie Vessel":POSITION 5,3:? #6;"CAN
NOT MOVE"
1460 REM All strings in 1470 in invers
e
1470 POSITION 3,5:? #6;"The Guardians"
:POSITION 5,6:? #6;"HAVE SAVED":POSITI
```

```
ON 2,7:? #6;"the human empire"
1480 POKE 764,255:GOTO 1530
1490 FOR J=1 TO 300:NEXT J:GRAPHICS 18
:POSITION 5,3:? #6;"THE MOTIES":POSITI
ON 3,4:? #6;"have conquered"
1500 REM 1st and last strings in 1510
in inverse char.
1510 POSITION 3,5:? #6;"THE GUARDIANS"
:POSITION 2,7:? #6;"THE HUMAN EMPIRE":
POSITION 4,8:? #6;"is doomed!"
1520 POSITION 1,10:? #6;"again?-PRESS
key":POKE 764,255
1530 IF PEEK(764)=255 THEN 1530
1540 GOTO 550
1550 REM Subroutine to draw galactic m
ap
1560 GRAPHICS 1:POKE 756,CSPAGE:DL=PEE
K(560)+PEEK(561)*256:FOR J=DL+7 TO DL+
15:POKE J,5:NEXT J
1570 J=J+1:IF PEEK(J)<>65 THEN 1570
1580 FOR K=0 TO 2:POKE DL+20+K,PEEK(J+
K):NEXT K
1585 POKE 708,LCOL:POKE 709,86:POKE 71
0,GCOL:POKE 711,MCOL:POKE 712,BKCOL
1590 POKE 87,0:FOR J=3 TO 7 STEP 2:FOR
 K=17 TO 24:POSITION K,J:? #6;"+":NEXT
 K:NEXT J
1600 FOR J=4 TO 6:POSITION 15,J:? #6;"
)":POSITION 25,J:? #6;")":NEXT J:FOR J
=2 TO 8:POSITION 20,J:? #6;")":NEXT J
1610 POSITION 18,4:? #6;",":POSITION 2
3,4:? #6;"*":POSITION 18,6:? #6;"*":PO
SITION 23,6:? #6;","
1620 POSITION 18,2:? #6;"*":POSITION 2
3,2:? #6;",":POSITION 18,8:? #6;",":PO
SITION 23,8:? #6;"*"
1630 FOR J=1 TO 11:X=ASC(XP$(J)):Y=ASC
(YP$(J)):POSITION X,Y:? #6;"#$":NEXT J
1640 POSITION 5,0:? #6;"Motie World":P
OSITION 5,11:? #6;"Human Empire"
1650 FOR J=1 TO 11
1660 IF I$(J,J)="g" THEN X=ASC(XP$(J))
:Y=ASC(YP$(J)):POSITION X,Y:? #6;"%&"
1670 IF I$(J,J)="m" THEN X=ASC(XP$(J))
:Y=ASC(YP$(J)):POSITION X,Y:? #6;"'(":
REM '( ARE IN INVERSE CHAR.
1680 NEXT J:RETURN
1700 GRAPHICS 10:POKE 560,LOSAV:POKE 5
61,HISAV:W1=ADR(W1$):W2=ADR(W2$)
1710 R=INT(RND(0)*16)*16:RESTORE 480:F
OR I=1 TO 8:READ A:POKE 704+I,A+R:NEXT
 I
1720 FOR J=150 TO 50 STEP -1:A=USR(W1)
1730 GOSUB 1770:NEXT J
1740 FOR J=50 TO 150:K=K+D:A=USR(W2)
1750 GOSUB 1770:NEXT J
1760 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND
 2,0,0,0:RETURN
1770 SOUND 0,J,4,12:SOUND 1,J+2,12,12:
SOUND 2,J+4,10,6:RETURN
1775 REM Move cursor subroutine (stick
 0)
1780 X=ASC(XP$(CURN+1)):Y=ASC(YP$(CURN
+1)):LOCATE X,Y,Z:LOCATE X+1,Y,ZZ:POSI
TION X,Y:? #6;CHR$(33);CHR$(34)
1790 FOR J=1 TO 20
1800 IF MFLAG=1 THEN SOUND 0,150+J,10,
10:SOUND 1,170-J,10,10:NEXT J
1810 POSITION X,Y:? #6;CHR$(Z);CHR$(ZZ
)
1820 S=STICK(0)
1830 IF (S=14 AND Y>3) OR (S=14 AND X=
20 AND Y=3) THEN Y=Y-2
1840 IF (S=13 AND Y<7) OR (S=13 AND X=
20 AND Y=7) THEN Y=Y+2
1850 IF S=11 AND X>15 AND Y>1 AND Y<9
THEN X=X-5
1860 IF S=7 AND X<25 AND Y>1 AND Y<9 T
HEN X=X+5
1870 FOR J=1 TO 11:IF ASC(XP$(J,J))<>X
 THEN NEXT J
1880 IF ASC(YP$(J,J))<>Y THEN NEXT J
1890 CURN=J-1
1900 IF STRIG(0)<>0 THEN 1780
1910 SOUND 0,0,0,0:SOUND 1,0,0,0:RETUR
N
1920 REM ** This subroutine reserves 5
 pages **
1930 REM **     at the top of RAM, mov
es the   **
1940 REM ** character set from ROM and
 changes **
1950 REM ** the characters ! thru , **
1960 DIM MCS$(42):RESTORE 1970:FOR J=1
 TO 42:READ A:MCS$(J,J)=CHR$(A):NEXT J
1970 DATA 104,169,0,133,203,133,205,16
9,224,133,204,165,106,56,233,5,133,106
,24
1980 DATA 105,1,133,206,162,4,160,0,17
7,203,145,205,200,208,249,230,204,230,
206,202,208,242,96
1990 A=USR(ADR(MCS$)):CSPAGE=PEEK(106)
+1:CS=256*CSPAGE
2000 RESTORE 2010:FOR J=CS+8 TO CS+103
:READ A:POKE J,A:NEXT J:RETURN
2010 DATA 0,0,2,9,39,9,2,0
2020 DATA 0,0,128,96,216,96,128,0
2030 DATA 10,37,154,152,152,154,37,10
2040 DATA 160,88,166,38,38,166,88,160
2050 DATA 15,61,55,247,247,55,61,15
2060 DATA 240,124,220,255,95,220,124,2
40
2070 DATA 60,220,215,221,223,220,220,6
0
2080 DATA 60,55,215,119,247,247,55,60
2090 DATA 0,2,1,2,1,2,1,0
2100 DATA 2,1,8,4,32,16,128,64
2110 DATA 0,0,0,24,0,0,0,0
2120 DATA 128,64,32,16,8,4,2,1
```

ERACE

Next meeting: 7.30 pm, April 5, 1983.
295 Foxtail Dr., Eugene, OR. 6 8 7 - 1 1 3 3 .

BROWN'S HAWKS

```
10 ? "}":DIM A$(5),B$(5),C$(5),D$(5),A
N$(1),Z$(5):Z$="      "
20 POKE 710,196:POKE 709,14:? :? ," HU
NGRY HAWKS ":? :? "THIS IS A ONE OR TW
O PLAYER GAME."
25 ? :? "EACH PERSON CONTROLS A HAWK :
-":? "PLAYER 1 CONTROLS HENRY CHICKEN
HAWK"
26 ? "PLAYER 2 CONTROLS HARRY HAWK-EYE
":? :? "HENRY IS BLUE & HARRY IS BROWN
"
30 POKE 752,1:? :? "PLAYER 1 USES JOYS
TICK 1":? "PLAYER 2 USES JOYSTICK 2":?
 :? "USE THE JOYSTICK TO MOVE SIDEWAYS
"
35 ? "AND PRESS THE FIRE BUTTON TO SWO
OP":? "DOWN ON THE PREY."
40 ? :? "PRESS THE START BUTTON TO STA
RT GAME":? :? " by  SYDNEY.H.BROWN  -
-   COMPCO    ";
70 IF PEEK(53279)()6 THEN 70
75 A$=" wvx ":B$=" yvz ":C$=" wvx ":D$
=" yvz ":POKE 106,PEEK(106)-2
80 GRAPHICS 2:POKE 710,0:POKE 708,14:P
OKE 709,40:POKE 711,222
90 ? #6;"ANIMAL":? #6;"frog      3 poi
nts":? #6;"rabbit    5 points":? #6;"t
urtle    1 point":? #6;"chicken   2 po
ints"
91 ? #6;"grub      2 points"
101 POKE 710,0:POKE 708,14:POKE 709,40
:POKE 711,222:A=PEEK(106)*256:FOR B=0
TO 511
102 IF B)431 THEN READ D:POKE A+B,D:NE
XT B:GOTO 110
106 POKE A+B,PEEK(57344+B):NEXT B
110 POSITION 0,8:? #6;"PRESS START BUT
TON"
111 IF PEEK(53279)()6 THEN 110
112 POSITION 0,0:? #6;")":POKE 710,140
:POKE 756,PEEK(106)
190 V1=1:V2=1:H1=1:H2=14:C1=0:C2=0:S1=
0:S2=0:NA=0
195 POKE 709,142:POKE 711,24:POKE 712,
128:POKE 710,200:POSITION 0,0:? #6;"sc
ore1=0  score2=0"
199 GOSUB 550
200 S=STICK(0):IF S=11 AND H1)0 THEN H
1=H1-1
201 IF S=7 AND H1(15 THEN H1=H1+1
202 IF STRIG(0)=0 THEN C1=1
205 POSITION H1,V1:? #6;A$;
210 ON R GOSUB 500,510,520,530,540
223 IF C2=1 THEN GOSUB 400:GOTO 235
224 IF C2=2 THEN GOSUB 450:GOTO 235
225 S=STICK(1):IF S=11 AND H2)0 THEN H
2=H2-1
226 IF S=7 AND H2(15 THEN H2=H2+1
227 IF STRIG(1)=0 THEN C2=1
230 POSITION H2,V2:? #6;C$;;
235 ON R GOSUB 500,510,520,530,540
248 IF C1=1 THEN GOSUB 300:GOTO 260
249 IF C1=2 THEN GOSUB 350:GOTO 260
250 S=STICK(0):IF S=11 AND H1)0 THEN H
1=H1-1
251 IF S=7 AND H1(15 THEN H1=H1+1
252 IF STRIG(0)=0 THEN C1=1
255 POSITION H1,V1:? #6;B$;;;
260 ON R GOSUB 500,510,520,530,540
273 IF C2=1 THEN GOSUB 400:GOTO 285
274 IF C2=2 THEN GOSUB 450:GOTO 285
275 S=STICK(1):IF S=11 AND H2)0 THEN H
2=H2-1
276 IF S=7 AND H2(15 THEN H2=H2+1
277 IF STRIG(1)=0 THEN C2=1
280 POSITION H2,V2:? #6;D$;
285 ON R GOSUB 500,510,520,530,540
297 IF C1=1 THEN GOSUB 300:GOTO 210
298 IF C1=2 THEN GOSUB 350:GOTO 210
299 GOTO 200
300 POSITION H1,V1:? #6;Z$:V1=V1+1:POS
ITION H1,V1:? #6;A$:IF V1(8 THEN RETUR
N
305 C1=2:LOCATE H1+2,V1+1,Z1:IF Z1=32
THEN RETURN
310 AC=1
350 POSITION H1,V1:? #6;Z$:V1=V1-1:POS
ITION H1,V1:? #6;B$:COLOR Z1:PLOT H1+2
,V1+1:COLOR 32:PLOT H1+2,V1+2:IF V1)1
THEN RETURN
355 C1=0:IF AC()1 THEN RETURN
356 SOUND 0,255,10,14:COLOR 32:PLOT H1
+2,V1+1:FOR W=1 TO 21:NEXT W:SOUND 0,0
,0,0
359 S1=S1+AS:POSITION 7,0:? #6;S1;:AC=
0:GOTO 550

400 POSITION H2,V2:? #6;Z$:V2=V2+1:POS
ITION H2,V2:? #6;C$:IF V2(8 THEN RETUR
N
405 C2=2:LOCATE H2+2,V2+1,Z2:IF Z2=32
THEN RETURN
410 AC=2
450 POSITION H2,V2:? #6;Z$:V2=V2-1:POS
ITION H2,V2:? #6;D$:COLOR Z2:PLOT H2+2
,V2+1:COLOR 32:PLOT H2+2,V2+2:IF V2)1
THEN RETURN
455 C2=0:IF AC()2 THEN RETURN
456 SOUND 0,123,10,14:COLOR 32:PLOT H2
+2,V2+1:FOR W=1 TO 21:NEXT W:SOUND 0,0
,0,0
459 S2=S2+AS:POSITION 17,0:? #6;S2;:AC
=0:GOTO 550
500 IF AC()0 THEN RETURN
501 IF X)19 THEN 550
502 IF FC=0 THEN POSITION X,9:? #6;AN$
503 FC=FC+1:IF FC/9()INT(FC/9) THEN RE
TURN
504 SOUND 0,255,10,14:FOR W=1 TO 10:NE
XT W:SOUND 0,0,0,0
505 COLOR 32:PLOT X,9:X=FC/3
506 IF X(20 THEN POSITION X,9:? #6;AN$
:RETURN
509 COLOR 32:PLOT X-3,9
510 IF AC()0 THEN RETURN
511 IF X)19 THEN 550
512 IF X=0 THEN POSITION X,9:? #6;AN$:
X=X+1:RETURN
513 XX=INT(RND(0)*3):IF XX=2 THEN XX=0
514 IF XX=1 THEN SOUND 0,14,10,10:FOR
W=1 TO 5:NEXT W:SOUND 0,0,0,0
515 POSITION X-1,9:? #6;" ";:POSITION
X,9:? #6;AN$:X=X+XX:RETURN
520 IF AC()0 THEN RETURN
521 IF X)19 THEN 550
522 IF X=0 THEN POSITION X,9:? #6;AN$:
X=X+1:RETURN
525 POSITION X-1,9:? #6;" ";:POSITION
X,9:? #6;AN$:X=X+0.25:RETURN
530 IF AC()0 THEN RETURN
531 IF X)19 THEN 550
532 IF X=0 THEN POSITION X,9:? #6;AN$:
X=X+1:RETURN
533 SOUND 0,7,8,10:FOR W=1 TO 3:NEXT W
:SOUND 0,0,0,0
535 POSITION X-1,9:? #6;" ";:POSITION
X,9:? #6;AN$:X=X+0.5:RETURN
540 IF AC()0 THEN RETURN
541 IF X)19 THEN 550
542 IF X=0 THEN POSITION X,9:? #6;AN$:
X=X+1:RETURN
545 POSITION X-1,9:? #6;" ";:POSITION
X,9:? #6;AN$:X=X+0.5:RETURN
550 SOUND 0,0,0,0:NA=NA+1:IF NA=51 THE
N 600
551 POKE 77,0:FC=0:COLOR 32:PLOT 19,9:
X=0:AC=0:R=INT(RND(0)*5)+1:ON R GOTO 5
56,552,553,554,555
552 AS=5:POKE 708,10:AN$="\":RETURN
553 SOUND 0,255,8,14:AS=1:POKE 708,22:
AN$="]":RETURN
554 AS=2:POKE 708,222:AN$="'":RETURN
555 SOUND 0,7,8,8:AS=2:POKE 708,56:AN$
="_":RETURN
556 AS=3:POKE 708,188:AN$="[":RETURN
600 POSITION 6,5:? #6;"THE  END":? #6:
? #6;" press start button":? #6:? #6;"
  [   \   ]   ^   _   "
605 IF PEEK(53279)()6 THEN 605

609 GOTO 111
1000 DATA 28,42,255,227,247,42,65,162,
6,15,31,61,48,32,64,0,48,120,252,222,1
34,2,1,128
1005 DATA 0,0,0,3,7,79,62,24,0,0,128,2
24,240,121,62,140
1010 DATA 0,4,10,63,126,252,226,121,0,
0,30,135,124,124,130,97,0,0,0,48,123,2
54,72,132
1020 DATA 4,11,4,252,124,56,8,20,10,14
,21,142,138,206,254,124
```

Game Writers' Column

by Jon Atack, Eugene A.C.E.

Welcome back to the GAME WRITERS' COLUMN. This month, due to a number of requests, we'll go over how to convert SynAssembler source code to Atari Editor/Assembler source code, and vice versa. I haven't used the Atari Assembler for some time now and am not overly familiar with it, so if I'm wrong about a conversion, please let me know so I can publish the correction.

Here is a list of the SynAssembler opcodes, and their equivalent opcodes on the Atari Assembler:

.AS daaa...ad
00100 .AS "THIS IS AN ASCII STRING"
The .AS daaa...ad directive translates the string of characters into their ASCII bytes. The string can contain any printing character and can be of any length.
I don't think there is any equivalent command on the Atari Assembler (I'm not positive though, so ask someone who knows just to be sure) except to translate the bytes yourself and to use a .BYTE directive. Example: .BYTE $34,$25,$26,$3F,$21 etc.

.AT daaa...ad
00100 .AT "THIS IS AN ATASCII STRING"

The .AT directive is the same as .AS except it translates into ATARI ASCII.

.BS exp
(any valid number or expression)
00100 .BS 44
00200 .BS $1C
Reserves a block of exp
bytes at the current location in the program. The expression just specifies the number of bytes to advance the location counter.
There is an equivalent opcode for the Atari Assembler, but I don't have the Atari manual and I haven't been able to find it. If there are only a few bytes you want reserved, then just use a .BYTE 0,0,0,0,0,0,0,0,0,0 for the number of bytes you want.

.DA expression or label (two bytes, LSB first)
.DA #exp or label (one byte, LSB of exp or label)
.DA /exp or label (one byte, MSB of exp or label)
The .DA data directive creates a constant or variable in your program. Here are some examples:
4000: 64 00 00100 .DA 100
4002: 64 00110 .DA #100
4003: 00 00120 .DA /100
4004: FF 3B 00130 .DA LOOP1
4006: FF 00140 .DA #LOOP1
4007: 3B 00150 .DA /LOOP1
Equivalent directives on the Atari Assembler are .BYTE (for .DA #exp), .DBYTE (for .DA exp), or .WORD (which outputs using MSB-LSB, instead of the normal LSB-MSB.).

.EN
00100 .EN
Tells the SynAssembler where to stop assembling the source program. .EN is purely optional if you want to assemble the entire program.
I think the equivalent for the Atari Assembler is .END.

label .EQ
expression

Defines the label to have the value of the expression.
00100 PMBAS .EQ $D400
00110 HPOS1 .EQ 53248
00120 STICK .EQ $278
The equivalent Atari Assembler directive is =.
00100 PMBAS = $D400
00110 HPOS1 = 53248
00120 STICK = $278

label .HS aabbccddeeff...
DATA1 .HS 3400AA4F039B001A
Stores the string of hex digits at the current location, two digits per byte. There must be an even number of hex digits in the string.
There is no equivalent command on the Atari Assembler except to use the .BYTE directive. Example: DATA1 .BYTE $34,0,$AA,$4F,$3,$9B,0,$1A

.LI OFF AND .LI ON
00100 .LI OFF
05680 .LI ON
These directives turn the assembly listing on and off, and can both be put anywhere in the program. The program assembles much faster without a listing, although the SynAssembler assembles so fast it usually makes no difference.
The equivalent Atari Assembler directives are .OPT LIS and .OPT NOLIS.

.IN [file name]
00100 .IN "D:SETUP"
00110 .IN "D:PART1.SRC"
00120 .IN "D:SCORING"
Causes the specified source file to be included in the assembly at the current location of the program counter. The program in memory at the time the ASM command is typed is called the "root" program. Only the root program may have .IN directives in it.
The .IN directive is useful for assembling very large programs which cannot fit in memory all at once. It is also very useful for connecting a library of subroutines with the main program.
I don't think there is an equivalent command for the Atari Assembler.

.OR
expression

00100 .OR $4000
02160 .OR START + $1C00
This sets the Program Origin and the Target Address to the value of the expression. Program origin is the address at which the object program will be executed. Target address is the address at which the object program will be stored during the assembly. The .OR directive sets both of these addresses to the same value, which is the normal way of operating.
For the Atari Assembler use * =. Example: * = $4000 or * = START + $1C00.

.TA
expression

00100 .TA $4C1F
Used the same as the .OR command, .TF sets only the Target Address at which the object code will be stored during assembly. Object code is produced and ready to run at the Program Origin, but is stored starting at the Target Address.
I don't think there is a similar function on the Atari Assembler.

.TF [file name]
00100 .TF "D:PROGRAM.OBJ"
Causes the object code to be stored on a binary file on disk rather than in memory. Only the code which follows the .TF directive will be stored on the disk.
To simulate this command on the Atari Assembler, use the ASM command as follows: ASM ,,#D:filename.

Other miscellaneous changes and differences:

1. For the SynAssembler, remove all references to "A". For example, in the instruction LSR A, the "A" should be removed. The SynAssembler assumes the "A" reference automatically. Likewise, the "A" is required for similar instructions on the Atari Assembler.

2. SynAssembler has no multiply or divide, so these must be put in by long hand.

3. To get HI and LO bytes, make the following changes:

Atari Assembler	LDA #PLACE/256	high byte
	LDA #PLACE/255	lo byte
SynAssembler	LDA /PLACE	high byte
	LDA #PLACE	lo byte

4. Finally, don't forget to change all the Atari directives to the above SynAssembler equivalents (or vice versa.)

5. Local labels on the SynAssembler (labels starting with a period, such as .1 or .23) must be changed to the regular Atari label format of capitals and numbers. On the SynAssembler, local labels can be used over and over (instead of making up a new name for every label) as long as they aren't used more than once between normal labels.

I hope this helps in converting source programs from one assembler to the other. Be watching for next month's column on machine language sound techniques, with some mind-blowing example programs in BASIC and machine language. Also, for you game-players, be ready for some hot new games coming out from Atari, Datasoft, and Synapse Software in the next few months. I am presently in the middle of a new game for Synapse that should hit the market sometime this summer — more on this in next month's column. Until then, good luck and great game-making!

— Jon Atack

FLASH!

The ATARI Service Contract, reported several months ago in ACE is now available through your localAtari Service Station. Costs $45 a year for the 400 and $120 for the 800. It includes coverage onall Atari made equipment -- disk drives, joysticks, cassette, printer, etc. Your computer must beworking at the time you sign up.

Hi-Res is a new magazine for the Atari looking for writers. Longwood Business Center, 755-B W. San Lando Springs Dr., Longwood, FL 32750

CALL TO ASSEMBLY

from S*P*A*C*E by Tom Newman

How to use pictures created with MICRO-PAINTER in your machine language programs (part two).

In part one of this article, which appeared in volume two issue 9 of the Space news letter (September 1982), I shared a program with you I had written to load MICRO-PAINTER files into a specified area of RAM, and display the pictures on the television screen. In this the second of three parts, I will show you how to compact the picture data and automatically save the condensed information to your disk drive for future use in a program. The first program was written with ATARI'S assembler/editor cartridge. The program included in this artlcle was written with the SYNASSEMBLER from SYNAPSE SOFTWARE. The SYNASSEMBLER is as much as fifty times faster than the assembler/editor cartridge, and supports all of the 6502 opperations, which the cartridge does not. Conversion between the two assemblers should not be difficult for anyone with even minimal experience in the use of such programs. Eg:

ATARI SYNASSEMBLER

* = ...-.OR

.BYTE $A0,$2D,$3E.....HS A02D3E

.BYTE "STRING".....—-.AS "STRING"

= .....—.EQ

This program was written for a 32K machine with at least one disk drive.

In any picture created by the computer, there will always be patterns repeated in either a vertical or horizontal direction. This program takes advantage of this fact and creates an inter-leaved table of data values and the number of times they are repeated, using a vertical scan of the screen refresh area. I have found through experimentation that a vertical scan results in the most efficient use of this type program. As explained earlier, MICROPAINTER uses antic mode 'E', which is a four color single line resolution mode. Each line contains forty bytes of data and each byte represents four pixels. Each pixel uses two bits to define its color and shape:

00 BACKGROUND COLOR

01 PLAYFIELD 0

10 PLAYFIELD 1

11 PLAYFIELD 2

Since each pixel requires two bits for color definition it is also twice as wide as it is high.

There are a total of 192 lines used to make up the picture, and 192 times 40 bytes per line equals 7680 bytes. This program should at least cut that in half for most pictures and do even better on simpler ones. I have included a couple of screens from the DONKEY KONG program my sons are working on to give you some idea of what I consider to be an average picture. Screen 1 was reduced to 3508 bytes, and screen 2 was reduced to 4057 bytes. The un-compaction program, which will be presented in part three occupies less than one page of memory, so the combination of controlling program and data base can be very cost effective in terms of memory useage. Even if you have only one playfield image this program will give you a 'blueprint' of the original image, eliminating the need for disk access each time the program is re-run. Now let's examine the program.

Lines 130 through **290** contain the equate table. The labels of the equate table allow us to use words to identify memory locations or values used in our programs, which we may have difficulty remembering. The labels also allow the assembler to keep track of where in memory things are, so we don't have to calculate their locations.

Lines 330 through **500** contain the initilization program. This is the part of the program which sets up all the starting values, such as data locations and counter initilization.

Lines 540 through **1090** contain the main body of the program. The program starts by getting the first byte from the screen refresh area and comparing it to each sucessive byte in the column. A counter called BLKSIZE keeps track of how many bytes have been examined up to a maximum of 256. When a mismatch occures or a count of 256 has been reached the program jumps to a subroutine called SAVEBLK and starts building the data base. SAVEBLK saves the data value first and then the number of repetitions indicated by BLKSIZE in encoded form. Any byte examined will be repeated at least once, but no more than 256 times. A single byte will hold numerical values from 0 to 255. Since nothing will ever be repeated zero times a literal use of the counter contents allows only 255 values to be stored. The answer to this inefficiency is found by complementing each bit of the the counter value. It is the complemented byte you will find in the table. The un-compacting program uses a counter which counts to zero, as an example:

$00 XOR $FF equals $FF, $FF counts to zero in one step, so $FF represents 1.

$FF XOR $FF equals $00, $00 counts to zero in 256 steps, so $00 represents 256.

The 'X' register is used to keep track of the line count. When it has reached 192 the column counter (the 'Y' register) is incremented and the home address is restored to the indirect buffer pointer. This is necessary because forty had to be added to buffer address each time a byte in the column was examined, but the starting address of each column is sequential. After all of screen RAM has been examined and the data base is built, the program waits for the user to press SELECT and OPTION keys simultaneously. It will then save the data base as a binary file using the filename 'COMPACT.TBL'.

Line 2000 contains the filename. You can insert any legal filename you like between the quotes. Remember to change this name if you want to save multiple compacted pictures on the same disk.

The created file contains all the proper header information of an ATARI binary file, and may be loaded into RAM at $4000 by DOS, or the ASSEMBLER/ EDITOR cartridge.

Lines 1140 through **1230** contain the program which waits for the user to press the SELECT and OPTION keys at the same time. The program also expects you to hold them down for about ⅓ second before acting.

Lines 1270 through **1430** set up the header bytes for the compacted file.

Lines 1470 through **1540** calculate the table length in bytes, which is needed by SIO for the save routine.

Lines 1590 through **2030** set up IOCB #1 for a byte aligned binary save:

1. IOCB #1 is closed 1590-1620

2. IOCB #1 opened for write 1660-1760

3. IOCB #1 executes the save 1800-1900

4. IOCB #1 is closed 1940-1960

In order to make use of this program you must first have the MICRO-PAINTER retrieval program assembled into memory.

Its execution point is at HEX 5F00. Remember to change the filename in the program to match the picture filename you are interested in. Now assemble the compactor program in this article and execute it at HEX 5C00. Wait a second, then press OPTION and SELECT. Disk drive one should come on and the compacted data base will be saved. That's all until next time. If you have any questions feel free to call. (206) 1-858-7649

```
00010          .LI OFF
00020          .OR $5C00
00030 ********************************
00040 *         COMPACT.ASM          *
00050 *  NEWSLETTER ARTICLE [PART2] *
00060 *         BY TOM NEWMAN        *
00070 *         NOVEMBER 1982        *
00080 *      S.P.A.C.E. Seattle,WA   *
00090 *                              *
00100 ********************************
00110 *         EQUATE TABLE         *
00120 ********************************
00130 BUFFER   .EQ $F0
00140 BUFFLO   .EQ $F0
00150 BUFFHI   .EQ $F1
00160 TABLE    .EQ $F2
00170 TBLLO    .EQ $F2
00180 TBLHI    .EQ $F3
00190 SAVEY    .EQ $F4
00200 REF      .EQ $F5
00210 HOMELO   .EQ $F6
00220 HOMEHI   .EQ $F7
00230 BLKSIZE  .EQ $F8
00240 COUNT    .EQ $F9
00250 HEADER   .EQ $FA
00260 LENLO    .EQ $FC
00270 LENHI    .EQ $FD
00280 NMIRES   .EQ $D40F
```

```
00290 CONSOL   .EQ $D01F
00300 ********************************
00310 *       INITILIZATION          *
00320 ********************************
00330 START    LDA #$50
00340          STA BUFFLO
00350          STA HOMELO
00360          LDA #$61
00370          STA BUFFHI
00380          STA HOMEHI
00390          LDA #$00
00400          STA COUNT
00410          STA TBLLO
00420          STA BLKSIZE
00430          TAY
00440          LDA #$FA
00450          STA HEADER
00460          LDA #$3F
00470          STA HEADER+1
00480          LDX #$C0
00490          LDA #$40
00500          STA TBLHI
00510 ********************************
00520 *        PROGRAM BODY            *
00530 ********************************
00540 BLOCK    LDA (BUFFER),Y
00550          STA REF
00560 NEXT     INC BLKSIZE
00570          BNE CONTINUE
00580          JSR SAVEBLK
00590 CONTINUE DEX
00600          BNE SKIPCOLM
00610          JSR NXTCOLM
00620          JMP CORRECT
00630 SKIPCOLM CLC
00640          LDA BUFFLO
00650          ADC #$28
00660          STA BUFFLO
00670          LDA BUFFHI
00680          ADC #$00
00690          STA BUFFHI
00700 CORRECT  LDA REF
00710          CMP (BUFFER),Y
00720          BEQ NEXT
00730          JSR SAVEBLK
00740          JMP BLOCK
00750 SAVEBLK  STY SAVEY
00760          LDY #$00
00770          LDA REF
00780          STA (TABLE),Y
00790          JSR STEPTBL
00800          DEC BLKSIZE
00810          LDA BLKSIZE
00820          EOR #$FF
00830          STA (TABLE),Y
00840          JSR STEPTBL
00850          LDA #$00
00860          STA BLKSIZE
00870          LDY SAVEY
00880          RTS
00890 STEPTBL  CLC
00900          LDA TBLLO
00910          ADC #$01
00920          STA TBLLO
00930          LDA TBLHI
00940          ADC #$00
00950          STA TBLHI
00960          RTS
00970 NXTCOLM  INY
00980          CPY #$28
00990          BEQ OUT
01000          LDA HOMELO
01010          STA BUFFLO
01020          LDA HOMEHI
01030          STA BUFFHI
01040          LDX #$C0
01050          RTS
01060 OUT      PLA
01070          PLA
01080          LDA REF
01090          JSR SAVEBLK
01100 ********************************
01110 *     WAIT FOR USER INPUT        *
01120 *  PRESS 'OPTION AND SELECT'     *
01130 ********************************
01140 USER     LDA CONSOL
01150          CMP #$01
01160          BNE USER
01170 WAIT     LDA NMIRES
01180          AND #$40
01190          BEQ WAIT
01200          INC COUNT
01210          LDA COUNT
01220          CMP #$14
01230          BNE USER
01240 ********************************
01250 *HEADER SETUP FOR BINARY FILE   *
01260 ********************************
01270          LDY #$00
01280          LDA #$FF
01290          STA (HEADER),Y
01300          INY
01310          STA (HEADER),Y
01320          INY
01330          LDA #$00
01340          STA (HEADER),Y
01350          INY
01360          LDA #$40
01370          STA (HEADER),Y
01380          INY
01390          LDA TBLLO
01400          STA (HEADER),Y
01410          INY
01420          LDA TBLHI
01430          STA (HEADER),Y
01440 ********************************
01450 *   ESTABLISH BUFFER LENGTH      *
01460 ********************************
01470          SBC #$40
01480          STA LENHI
01490          CLC
01500          LDA TBLLO
01510          ADC #$07
01520          STA LENLO
01530          LDA LENHI
01540          ADC #$00
01550          STA LENHI
01560 ********************************
01570 *       CLOSE IOCB #1            *
01580 ********************************
01590          LDX #$10
01600          LDA #$0C
01610          STA $342,X
01620          JSR $E456
01630 ********************************
01640 *   SETUP IOCB #1 FOR WRITE      *
01650 ********************************
01660          LDA #$03
01670          STA $342,X
01680          LDA #FILE
01690          STA $344,X
01700          LDA /FILE
01710          STA $345,X
01720          LDA #$08
01730          STA $34A,X
01740          LDA #$00
01750          STA $34B,X
01760          JSR $E456
01770 ********************************
01780 *      GO WRITE THE FILE         *
01790 ********************************
01800          LDA #$0B
01810          STA $342,X
01820          LDA #$FA
01830          STA $344,X
01840          LDA #$3F
01850          STA $345,X
01860          LDA LENLO
01870          STA $348,X
01880          LDA LENHI
01890          STA $349,X
01900          JSR $E456
01910 ********************************
01920 *       CLOSE IOCB #1            *
01930 ********************************
01940          LDA #$0C
01950          STA $342,X
01960          JSR $E456
01970 ********************************
01980 *    FILE NAME FOR TABLE         *
01990 ********************************
02000 FILE     .AS "D:COMPACT.TBL"
02010          .HS 9B
02020          BRK
02030          .END
```

MARCH MEETING
WILL BE HELD AT
THE L.C.C. CAMPUS
IN
FORUM 308

Wed. Mar 9 at 7:30 pm
FORUM is on the SOUTH
side of campus. The
FORUM bldg is near the
middle of the south
parking lot area.

ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

FIRST CLASS MAIL

Atari Computer Enthusiasts

A.C.E. is an independent computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are $10 a year for U.S., and $20 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.

Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405
President—Kirt Stockwell, 1810 Harris #139, Eugene, Or 97403 / 503-683-3005
Vice Pres—Larry Gold, 1927 McLean Blvd., Eugene, Or 97405 / 503-686-1490
Secretary—Charles Andrews, POB 1613, Eugene, Or 974401613 / 503-747-9892
Librarian-Chuck and Jody Ross, 2222 Ironwood, Eugene, OR 97401 / 503-343-5545
Editors—Mike Dunn, 3662 Vine Maple Dr., Eugene, Or 97405 / 503-344-6193
Jim Bumpas, 4405 Dillard Rd., Eugene, Or 97405 / 503-484-9925
E.R.A.S.E. (Educational SIG Editor)—Ali Erickson, 295 Foxtail Dr., Eugene, Or 97405 / 503-687-1133

Send a business-size SASE to the Ross' for the new, updated ACE Library List!!

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of a Tara equipped 48K Atari 400, 2 double-density disk drives, an Atari 825 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband. See the Nov '82 issue for complete details.